Data Mining Applications of Singular Value Decomposition

Miklós Kurucz

Supervisor: András A. Benczúr Ph.D.

Eötvös Loránd University
Faculty of Informatics
Department of Information Sciences

Informatics Ph.D. School
András Benczúr D.Sc.

Foundations and Methods of Informatics Ph.D. Program
János Demetrovics D.Sc.

Budapest, 2011

Acknowledgment

I would like to express my deepest gratitude to my supervisor András Benczúr for all the support he gave me to do great research. I would like to thank him for all the fruitful conversations we had on spectral clustering and recommender algorithms. I will always remember with joy the moments when our solutions started to work. I have been grateful to him ever since he accepted me to write my M. Sc. thesis under his supervision and set me on the path I still follow.

I would also like to thank all my co-authors for their highly appreciated contribution to my research. Károly Csalogány who showed me how to put into practice all the theoretical knowledge I had. Balázs Rácz who made me understand what high quality software means. László Lukács and István Nagy for their excellent insights on various problems.

I would like to thank my fellow co-workers from room 407, István Bíró, Bálint Daróczy and Miklós Erdélyi for the great atmosphere they created, I value their continued friendship above everything else.

I would also like to thank my family, friends, present and past co-workers who made the creation of this thesis possible.

Contents

Chapter 1

Introduction

„I think it is not entirely unuseful to treat briefly a few of these problems [...] without which this most beautiful branch of mathematical science would remain confined ..."

Eugenio Beltrami

1.1 Overview

In 1873 Eugenio Beltrami described a new approach [21] to his students for the treatment of bilinear functions. This new method was in fact a special case of Singular Value Decomposition (SVD). Through the following years many more detailed results were published, most notably Karl Pearson's discovery of Principal Component Analysis (PCA) [135] in 1901, and the first complete proof of existence [62] by Carl Eckart and Gale Young in 1936.

Since Pearson's results there has been a growing interest to use these techniques on large datasets. Practical problems such as transportation of goods or minimalization of production costs lead to linear equations that can be solved efficiently by the usage of Generalised Inverse [137] (Penrose 1955). By 1970 the algorithms [79] used to compute the decomposition of matrices reached a level of sophistication that is taught to interested students of every quantitative science. Today SVD and PCA are amongst the most frequently used tools to solve statistical, signal processing and modeling tasks.

One of these tasks is Social Network Analysis, a very popular research area for a long time. Identifying groups, finding similar people, and observing social dynamics have all been widely researched. This information is very useful for marketing purposes, such as finding target groups for a new service, or identifying people who will likely stop using a service. As the amount of available data grew in recent years, new methods to extract information started to become popular. One such method is the spectral clustering, which has been reinvented after a few decades of disinterest. Spectral clustering uses SVD to find minimal cuts in networks.

Although spectral clustering is reported by many [54, 117, 166] as an efficient method to

extract user groups from graphs, it was known to fail for large power law graphs with several partly successful attempts[109].

Another area of research where SVD is extensively used is the prediction of people's interests. Finding products that are interesting for customers is becoming more and more important. When text descriptions of products are available, this can be achieved via simply finding items (by looking for the words) that usually interest the customer.

However, new content services consider the convergence of television, Internet and mobile devices, and focus on media content rather than on text. Recommendation technologies may add extra value to these services. Nowadays most of the TV and radio programs are accessible on the Internet, videos can be downloaded. This blurs the border between online programs and the Internet.

One might try to solve this issue by looking for people with similar taste, but naive approaches such as computing the whole similarity matrix will fail, due to the size of matrix. This problem can also be addressed with approximation algorithms.

Sparsity of the observed data is a known problem in recommender systems: users evaluate only a few of the objects with possible interest for them. While SVD may in some sense solve this problem, we believe that sparsity still causes problems for the following reason. SVD is an optimal low rank approximation in the Frobenius norm, i.e. the sum of the squared difference between the original and low rank estimated matrix entries. This rewards replacement of unknown values by fairly meaningless averages from large user groups instead of meaningful values.

It is important to note that for both problems our experiments were performed on real data sets. Our results on network clustering and characterization differ from prior work in two aspects. Firstly, in our evaluation we deployed external sociodemographic parameters such as geographic location in addition to graph properties. Secondly, our problems are larger, sometimes by several orders of magnitude than previously reported ones: our graph has nearly 50 million edges, which poses challenges even for more recent algorithms. Improved hardware capabilities require new algorithms and lead to new empirical findings in our work.

One major data set we used is the call graph of more than two million Hungarian landline telephone users [22], a unique data of long time range with sufficiently rich sociodemographic information on the users. Telephone call graphs are also used by the graph visualization community: [159] reports visualization on graphs close to the size of ours with efficient algorithms to select the neighborhood subgraph to be visualized. In addition [49] gives an example of long distance telephone call fraud application by manual investigation.

We included measurements on the LiveJournal blogger network where we showed the hardness of the spectral clustering task and also identified the well-known Russian user group [81, 165]. Prior to our work, this was the only known large scale formation of the LiveJournal

blogger network [81, 165]. By our methods we revealed clusters arranged by location, age and certain types of interest such as religion. We also partitioned the UK2007-WEBSPAM host graph where our algorithm was able to identify meaningful clusters while baseline algorithms completely fail.

For our experiments with recommendation algorithms we used the data provided for the famous $1,000,000 Netflix Prize Contest. Netflix made over 100 million ratings available from over 480 thousand randomly-chosen, anonymous customers on nearly 18 thousand movie titles [25] that we used to justify our SVD based recommender technique.

1.2 Our contribution

In our thesis we discuss our results and experiments with SVD. Firstly, we compare existing spectral clustering algorithms on large datasets. Secondly, we give pre- and postprocessing heuristics that increase clustering quality. Thirdly, we introduce a network model that explains the difficulty of spectral clustering. Finally, we show the applicablity of SVD on matrices with missing values.

Comparative analysis of various spectral clustering algorithms

We compared two different relaxation of the graph minimal cut problem. We measured algorithms from both branches of spectral clustering, the first one using several singular vectors, the other one using only the second (Friedler) vector. We also experimented with both the Laplacian and the weighted Laplacian matrices.

Pre- and postprocessing heuristics

We gave two preprocessing and one post-processing heuristic that solve the spectral clustering hardness problem. As for preprocessing we (1) contracted tentacles, long, loosely connected regions that add noise to the methods; and (2) identified and removed dense regions that distract too many eigenvectors. The dense regions were added back in the postprocessing phase: they were distributed along with the disconnected components of the clusters. For postprocessing we enforced a component size balance and redistributed disconnected parts of the resulting clusters.

Spectral Clustering hardness

Recently it was shown that spectral clustering fails to find balanced cuts in large social networks as it tends to chop off tentacles from the graph. The existing network models [18, 97, 100] fail to explain this behavior: Even when these models generate graphs with tentacles, spectral clustering gives meaningful cuts. This is in contrast with our observation for real networks.

We gave a new network model which has the same properties as real-world graphs, from the perspective of graph clustering.

Factorization of matrices with missing values

We calculated SVD both in an expectation maximization framework and in the context of the Lanczos algorithm. We calculated initial singular vectors by inputting some preset values for the missing element of the matrix. This input can be the global average or other meaningful values. In the following iterations we used the low-rank approximation matrix from the previous iteration as input. In order to speed up calculation we implemented two different convergence boosting techinque. We tested our methods on the Netflix data set.

1.3 Organization

Our thesis is organized as follows. Each chapter begins with a section introducing the problems to be discussed. Before giving the details of our work we list the relevant related results. Finally, we conclude each chapter with a short summary and possible directions for future research.

In the second chapter, we introduce the problems that can be solved by SVD, or where existing solutions can be improved with it.

The topic of the third chapter is the datasets we used for the various social network analysis and other network based tasks.

The fourth chapter the existing spectral clustering algorithms are described, and we show that how proper pre- and postprocessing algorithms improve clustering quality.

In the fifth chapter describes the network models currently used for most problem simulation. We examine these model from the viewpoint of spectral clustering, and we give a new model that explains that clustering hardness of real-world networks.

The sixth chapter contains our work on recommendation systems. First we briefly summarize our result on the KDD Cup 2007 competition. The rest of the chapter presents the applicability of SVD to matrices with missing values, and methods to speed-up computation.

In each section that describes our new contribution, the last section concludes with bibliographic notes added, including the original publications where the results first appeared, my contribution, as well as the list of papers citing those results.

Chapter 2

Applications of the Singular Value Decomposition

The Singular Value Decomposition (SVD) is a method to obtain the best rank k approximation of a matrix in the Frobenius as well as ℓ_2 norms, a method that can be applied both to reduce storage space for the matrix as well as to remove noise from the data. Its prominent applications include recommendation systems [57], information retrieval via Latent Semantic Indexing [27, 50, 133], Kleinberg's celebrated HITS algorithm for web search [2, 98], clustering [58, 116], and learning mixtures of distributions [1, 93] just to name a few. Classification can be solved by regularized regression [60] and text database querying by matrix-vector products [47].

Let us imagine that we are given n data points and each point has m attributes described by an $m \times n$ matrix of reals A where the i^{th} column of A represents the i^{th} data point. Our intuition is that while the algebraic rank of A might be as high as $\min\{m, n\}$, most of the information in A can be described as a linear combination of k latent factors. Our assumption is that the true "meaning" of A can be captured by a rank-k matrix A_k and that the error term $A - A_k$ is mostly due to irrelevant factors or noise in the data. In other words, we seek the best k-dimensional approximation of A which provides a form of lossy data compression and noise reduction [13, 133]. It is natural to measure the quality of a reduced rank approximation matrix B by the norm of the error term $C = A - B$. In particular, given an $m \times n$ matrix C, let $\|C\| = \max_{\|x\|=1} \|Ax\|$ denote the *spectral* and $\|C\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^2}$ the *Frobenius* norms of C.

Theorem 1 (Singular Value Decomposition [80]) *Let A be an $m \times n$ matrix with rank ρ. Then there exist matrices $U \in \mathbb{R}^{m\times\rho}$, $\Sigma \in \mathbb{R}^{\rho\times\rho}$ and $V \in \mathbb{R}^{n\times\rho}$ such that*

- *$A = U\Sigma V^T$,*
- *U and V are orthogonal, i.e. $U^TU = V^TV = I_\rho$, and*

- Σ *is diagonal and* $\Sigma_{11} \geq \Sigma_{22} \geq \ldots \geq \Sigma_{\rho\rho} > 0$.

For $1 \leq i \leq \rho$ *we call the* i^{th} *column of* U *and* V, u_i *and* v_i, *the left and right singular vectors corresponding to the singular value* $\sigma_i = \Sigma_{ii}$.

Corollary 2 *Using the above notation we also have that*

- $\|A\| = \|Av_1\| = \left\|u_1^T A\right\| = \sigma_1$,
- $\|A\|_F = \sqrt{\sum_{i=1}^{\rho} \sigma_i^2}$.

Corollary 3 (Eckart-Young theorem) *For* $1 \leq k < \rho$ *let* $U_k \in \mathbb{R}^{m \times k}$ *and* $V_k \in \mathbb{R}^{n \times k}$ *contain the first* k *columns of* U *and* V *and let the diagonal* $\Sigma_k \in \mathbb{R}^{k \times k}$ *contain first* k *entries of* Σ. *We define* A_k, *the rank-*k *truncated SVD as* $A_k = U_k \Sigma_k V_k^T$. *Then*

- $\|A - A_k\|_F = \min\{\|A - B\|_F : \textit{rank}(B) \leq k\} = \sqrt{\sum_{i=k+1}^{\rho} \sigma_i^2}$,
- $\|A - A_k\| = \min\{\|A - B\| : \textit{rank}(B) \leq k\} = \sigma_{k+1}$.

Thus the best rank-k approximation of A with respect to both the Frobenius and spectral norm is given by the rank-k truncated SVD $A_k = U_k \Sigma_k V_k^T$. Recalling that the columns of A contain the data points we may interpret the columns of U_k as the latent factors or topics each with an importance of σ_i. The j^{th} column of V_k^T is the compressed representation of the j^{th} original data vector containing its coordinates or mixing weights for each latent factor.

For further details about the SVD and relevant linear algebra we refer the reader to [67, 80].

When experimenting with spectral clustering and other SVD based algorithms, we may put properties of the core SVD algorithm into new light by its interaction with the caller algorithm. We investigated the effect of the precision of the approximation, the number of dimensions used together with the density based clustering algorithms and its parameters (as e.g. k in k-means).

2.1 Spectral Graph Partitioning

Spectral clustering refers to a set of heuristic algorithms, all based on the overall idea of computing the first few singular vectors and then clustering in a low-dimensional (in certain cases simply one-dimensional [64]) subspace. The applicability of spectral methods to graph partitioning was observed in the early 70's [55, 64]. The methods are then rediscovered for netlist

partitioning, an area related to circuit design, in the early 90's [8, 9, 10, 40]. Since the "Spectral Clustering Golden Age" [54, 117, 166, etc] 2001 we only list a random selection of results. Spectral clustering is applied for documents [42, 43, 166] as well as image processing [113, 117, 147], see many earlier references in [92]. More recently, several approximate SVD algorithms appeared [58, 73, 142, and many others]; with the expansion of available data volumes their use in practice is likely in the near future.

The core idea is to relax the standard Quadratic Integer Program for graph bisection, $\frac{1}{4}x^T Lx$, where L is the graph Laplacian and x is the ± 1 cut indicator vector. In order to avoid the trivial cut with all nodes on one side, we have $x^T e = 1$ where e is a vector of all ones. When relaxing x to arbitrary real values between -1 and +1, the optimum is known to be the second eigenvector (the Fiedler vector) of L [64]. When however we relax indicator values to be arbitrary n-dimensional vectors of norm 1, the resulting optimization problem can be solved by semidefinite programming [109].

Variants of spectral partitioning dating back to the 1970's fall into two main branches as described in [10]. The first branch is initiated by the seminal work of Fiedler [64] who separates data points into the positive and negative parts along the principal axes of projection. His original idea uses the second singular vector, the so-called Fiedler vector; later variants [8, 19] use more vectors. Hagen and Kahng [86] are perhaps the first to use the second smallest eigenvalue for graph partitioning of difficult real world graphs.

The second branch of hierarchical spectral clustering algorithms divides the graph into more than two parts in one step. While the idea of viewing nodes as d-dimensional vectors after projecting the adjacency matrix into the space of the top k singular vectors is described already by Chan et al. [40], much later Zha et al. [166] introduce the use of k-means over the projection. In our recent work [104] we compare the two branches and find the superiority of using several spectral directions; in the rest of this section we restrict our attention to this class of algorithms.

Experiments on spectral graph partitioning either use the unweighted or weighted Laplacian as the input to the singular value decomposition procedure. The *Laplacian* is defined as $D - A$, where D is the diagonal matrix whose i-th entry is the total edge weight at node i and A is the adjacency matrix. The *weighted Laplacian* $D^{-1/2}AD^{-1/2}$ is first used for spectral bisection in [54, 147]. The Laplacian arises as the relaxation of the minimum ratio cut [86]; the weighted Laplacian appears in the relaxation of the normalized cut [147] and the min-max cut [54] problems. Weighting strategies are discussed in more detail in [10, and references therein], and Alpert and Kahng [8] empirically compared some of them. Unfortunately these results deal with the netlist partitioning problem only. Since netlists are hypergraphs, we may not directly use the findings of [8] but have to remember the importance of comparing different graph weighting strategies.

Clustering covers a wide class of methods to partition a set of data in order to locate relevant

information by grouping and organizing similar elements in an intelligible way. Since telephone companies have high quality data, their networks are ideal for experimenting. The purpose of clustering telephone users includes user segmentation, selection of communities with desired or undesired properties as e.g. high ADSL penetration or high recent churn rate. In a survey Newman [122] observes that in social network research "particular recent focus has been the analysis of communities".

The formation of the input matrix to SVD computation from the detailed call list strongly affects the outcome of clustering. Kannan et al. [92] suggest modeling the input as a similarity graph rather than as a distance graph, raising the question of how to turn the call information including the number, total duration and price between a pair of callers into a similarity measure. In addition to various ways of using cost and duration the node similarity measures of Section 3.2 may also be used to reweight the input graph.

2.2 Latent Semantic Analysis

LSA is a theory and method for extracting and representing the contextual usage meaning of words by statistical computations applied to a large collection of documents (corpus). It was introduced in 1988 by Deerwester et al. [50].

In this model a document is represented as a vector where each dimension corresponds to a separate feature from the document. A feature could be a term or any other unit that is a representative attribute of the documents in the given corpus. If a feature occurs in the document, its value in the vector is non-zero.

An important step in LSA is to transform the term-document vector space into a concept-document and document-concept vector space. By reducing the number of concepts, the documents and their terms are projected into a lower-dimension concept space. As a consequence, new and previously latent relations will arise between documents and terms. In order to apply LSA we first generate a term-document matrix $\mathbf{D}$ from the given corpus. Then, the singular-value decomposition (SVD) is applied to $\mathbf{D}$.

$$D = U\Sigma V^T, \tag{2.2.1}$$

Geometric interpretation The rows of the reduced matrices, U_k and V_k as in Corollary 3 respectively are taken as coordinates of points representing the documents and terms in a k dimensional space. With appropriate rescaling of the axes, by quantities related to the associated diagonal values of Σ, dot products between points in the space can be used to compare the corresponding objects.

Using the SVD decomposition, one can compare two terms, two documents or a document

with a term, the figures are similar, for example in document comparison:

$$D_k^T D_k = V_k \Sigma^2 V_k^T \tag{2.2.2}$$

matrix contains the document-to-document dot products.

One important feature of LSA is the generalization to unseen objects, i.e. one can compute the representation of objects, that did not appear in the original analysis. Let us say we are given a query expression composed of terms from the vocabulary. Using linear algebra, it is easy to show that the query can be represented as the centroid of its corresponding term points.

To sum up, the main advantages of LSA are:

- *Synonymy:* Synonymy refers to the fact that two or more different words have the same or similar meaning, such as *movie* and *film*. A traditional vector space model based Information Retrival (IR) system cannot retrieve documents discussing the topic of a given query unless they have common terms (due to the limitation of exact matching) however mapping the query and the document to the concept space, they are both likely to be represented by a similar weighted combination of the SVD variables, hence the cosine of the two vectors can be small.

- *Polysemy:* Polysemy refers to the fact that one word have multiple meaning, such as the word *bank*. The precision of the retrieval can be reduced significantly, if the query have a large number of polysemous word. Applying LSA to the query the rare and less important usages of certain terms can be filtered out, thereby increasing the precision of the search.

- *Term dependence*: The vector space model relies on the bag-of-words concept, i.e. the terms constituting the documents are completely independent from each other (they are orthogonal basis vectors of the vector space), however it is well known that there are strong correlations between terms. Term associations, for example can be exploited by adding phrases composed of two or more words to the vocabulary. LSA offers a more intuitive solution through the embedding of word-word, document-document and word-document correlations into the reduced LSA factor based representation. Note that, it comes at the price of an increased computational cost, this is only a one-time cost because one can build the LSA representation for the entire document collection once (i.e. performance at retrieval time is not affected).

and the main disadvantages of LSA are:

- *LSA and normally-distributed data:* As noted earlier, SVD finds a k-dimensional approximation of the term-document matrix (say A_k), given that the Frobenius-norm of the error term $(A - A_k)$ is minimized, or in other terms beside least-squares error. But, least-squares error are in fact suitable for normally-distributed data, not for count data as in the

term-document matrix. A possible solution is to use $tf \times idf$ weighting before appling SVD [114].

- *Storage:* it seems to be antinomic, but the space requirement of an SVD representation in several real datasets is larger than the sparse representation [89].

- *Efficiency:* Using vector space representation, one can build an inverted index for the documents, i.e. a table where the keys are the words, and the contents are the documents containing the key-word. Consequently, only documents that have some terms in common with the query must be examined during the retrieval process, however in the LSA representation, the query must be compared to every document in the collection.

2.3 Recommendation Systems

Recommendation systems aim at predicting the opinion of a certain user whether a given object is relevant, interesting or preferred. For instance, an application of very high importance is to recommend the advertisements best matching a given content, which is the basis for the business success of Google in connection with search results.

Applications of recommendation systems include building shared user models that are able to recognize the same client through signing in using different technologies; estimating the probability of a certain user buying a given product or liking a given media content; recommending customers products they most likely prefer; recommending products for a certain online customer that either belong to the category just being searched for, or are similar to the product just purchased, and to construct personalized view of newsletters and online content.

The recommendation problem can be formulated as follows: Let U be the set of all users, and I the set of all items that can be recommended. Let f be a function that measures the preference of an item for a user. $f : U \times I \rightarrow R$, where R is a totally ordered set. For each user $u \in U$ we want to recommend the best item $i' \in I$, formally:

$\forall u \in U, i'_u = \underset{i \in I}{\operatorname{argmax}}\, f(u, i)$.

In recommender systems there can be many kinds of preference functions. For example:

- There exists a user given rating like the star system on many online site.

- A rating is computed based on user-behavior, like the amount of time the user spent on with an item.

- In a profit-based system the price of items are also taken into account.

The central problem of recommendation system is that the goodness function f is not definied on the whole $U \times I$ matrix, but only on a part of it. This means that f has to be

extrapolated to the whole matrix. Once the unknown goodness is estimated, we recommend the best N items for the users.

Recommender algorithms are classified, based on how recommendations are computed:

- *Content-based recommendations:* The user will be recommended items similar to the ones the user preferred in the past.
- *Collaborative recommendations:* The user will be recommended items that people with similar tastes liked in the past.
- *Hybrid approaches:* Combination of the previous two.

There are some problems that recommenders usually have to deal with.

- *New user problem:* A user profile or history has to be created before accurate recommendations can be given.
- *New item problem:* Collaborative systems cannot make recommendation to items, not yet rated by some users.
- *Abundance of items:* The best items has to be choosen from thousands of items, computing goodness for all items is slow.
- *Limited content analysis:* Content-based techniques can only work on features explicitly associated with items. For items like movies these features can be hard to calculate.

The most common method for giving recommendations is the neighborhood based approach, also known as kNN(k Nearest Neighbors). It identifies similar users or items, and combines the goodness of these neighbors to get the unknown value.

Another approch is the matrix factorization method. The goal of these methods to uncover latent features that explain the goodness function. The methods to compute the features include: SVD, PCA, Non-negative Matrix Factorization.

Applying an SVD-based technique raises unique difficulties due to the sparsity issue. The conventional SVD computation requires that all entries of the matrix are known. In fact, the goal of SVD is not properly defined when some entries are missing.

Chapter 3

Exploration of Real Life Social Networks

There has been a considerable growth of interest in the properties of networks with a particular focus on the evolution of the contacts, the analysis of communities within networks or the classification of network objects. Networks underline all aspects of our life including friends, social contacts, computers and even brain cells or protein interaction in bacteria. Several surveys cover recent results: Barabási [15], Newman [121] or Scott [146] to name a few.

The purpose of investigating, measuring and modeling social networks may include the analysis of community formation or information spread within the network. The network of telephone communication contacts is particularly important in practice. Telephone call network models may serve the purposes of user segmentation or the selection of communities with desired or undesired properties. A desired community may be one with high ADSL penetration where new ADSL lines or other advanced services are likely sold with success to those members who have no subscription yet. An undesired community may be one with high recent churn rate where a campaingn may have to be designed to keep the members in the service. Other applications include viral marketing analysis [140] and other means of enhancing marketing communication by also relying on the spread of information within the social network.

In this chapter we introduce the three main data sets used in our experiments as well as explore the properties of these real life networks. We experiment with the LiveJournal Friends network, the call graph of Hungarian Telekom, and also the linkage of a 100,000-site crawl of the `.uk` domain. As an introduction, we perform the measurements of [111] and [97] over the telephone call graph. First in Section 3.2 we compare measures for the strength of the connection between members of the network by performing a link prediction experiment. Then in Section 3.3 we investigate the geographic location as a predictor of proximity in the social network.

Our experiments are performed on the LiveJournal Friends of more than three million users, where large scale methods are required to mine the latent information within the network. Compared to other networks such as the post network, this network is more robust in time and our

methods give access to patterns persistent on longer time scale.

The telephone graph appears less frequently in publications of the data mining community compared to the social network of bloggers [101, and references therein] or the World Wide Web [53, and many others]. Few exceptions include a theoretical analysis of connected components and eigenvalues [6, 45, 46] and several machine learning methods for churn prediction on real data [12, 157, etc.]. Closest to our results are the structural investigations of mobile telephone call graphs [119, 125, 126] and the sketch-based approximate k-means clustering of traffic among AT&T collection stations over the United States [48]; for the latter result however the underlying graph is much smaller (20,000 nodes) and the main goal was to handle the time evolution as an additional dimension.

In general we observe a strong topdown regional structure with large cities appearing as single clusters. These small world power law graphs are centered around very large degree nodes that distort clustering structure and must be removed prior to clustering to get any usable information even in the subgraph of private users. The heavily interconnected clusters are very hard to split further and bottom-up approaches for clustering get easily confused in their neighborhood. We find large fraction of users belonging to tentacles near community centers that require special pre and postprocessing in clustering algorithms that may efficiently build on breadth-first heuristics.

We compare measures for the strength of the connection between members of the network by performing a link prediction experiment. We investigate the geographic location as a predictor of proximity in the social network. We describe the characteristics of clusters that can be algorithmically found by measuring both graph properties and external sociodemographic parameters such as geographic location.

3.1 Data sets

Before presenting the main results, we describe the data sets that we use for illustration and show their main graph theoretic parameters. Our graphs obey the generally observed properties of social networks: they have power law degree distribution [17, 18, 31], small diameter [7, 156] and consist of a single giant component [6].

3.1.1 LiveJournal Friends network

For our experiments we use the LiveJournal friends network downloaded in a two-week period of November 2007[1]. The total number of users is 3,583,332 with 44,913,072 directed edges, out of which 14,286,827M is reciprocal. Another data set of Backstrom et al. [14] has 4.2M

[1] Available for research upon request from the author.

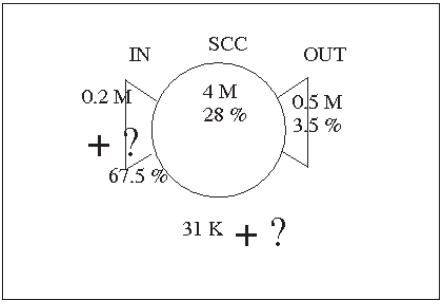


Figure 3.1: The Bow Tie structure of the LiveJournal friends network.

Table 3.1: Availability of metadata over the LiveJournal friends network.

Country	Age	Interest	School
76.03	39.79	62.82	47.31

users with no major reason for difference between the two collections. By manual analysis we observed certain users missing due to timeouts, some users renamed, also some friends changed. The union of the two collections has 4,720,668 users, less than 28% of the 14 million listed by LiveJournal as of November 2007.

Since we downloaded the Friends network starting from a single user, our collection consists of a giant strongly connected component (SCC) as well as nodes reachable from the SCC (OUT). The collection of [14] is started from community listings, hence the union of the two data sets partly reveal the bow-tie structure of Fig. 3.1 with 197,325 nodes not reachable from SCC but from which SCC can be reached (IN), and 31,157 users either disconnected (ISLAND) or reachable from IN or reach OUT but not in IN or OUT (TUNNEL). The number of strongly connected components is 768351 with a single giant one, leaving tiny pieces for others. The bow tie structure observed first for Web pages by [37], then by [168] for mailing lists is depicted in Fig. 3.1; the relation of the size of the strongly and weakly connected component of the post network is also described by [148].

In our analysis below we rely solely on our crawl since no user data is collected by [14]. We keep only bidirectional edges; this procedure leaves us with a giant component with 2,379,267 nodes and 14,286,827 reciprocal edges. Since graph partitioning requires a connected graph, we discard all other nodes.

The available metadata provided via a LiveJournal XML interface and the percentage of users who provide the information is summarized in Table 3.1 with a list of characteristic country locations in table 3.2.

Table 3.2: Top list of country location.

Country	Number	% known	% all
US	1 463 654	76.9	40.9
CA	87 609	4.6	2.4
RU	82 801	4.3	2.3
UK	73 789	3.8	2.1
AU	32 508	1.7	0.9
SG	14 986	0.7	0.4
DE	11 329	0.6	0.3
PH	10 380	0.5	0.3
UA	10 260	0.5	0.3
JP	7 778	0.4	0.2
FI	7 104	0.4	0.2
NL	5 970	0.3	0.2
NZ	4 958	0.3	0.1
FR	3 747	0.2	0.1

3.1.2 Telephone Call Graph

Our second data set consists of the telephone call graph of the Hungarian Telecom used in [104]. The telephone call graph is formed from the call detail record, a log of all calls within a time period including caller and callee id, duration, cost and time stamp. The vertex set consists of all nodes that appear at least once as caller or callee; over this set calls form directed edges from caller to callee. Edges are weighted by various aggregates of call multiplicity, duration or cost; time stamps are ignored in this work. The resulting graph obeys the power law degree distribution and contains a giant connected component of almost all nodes [6]. For a time range of 8 months, after aggregating calls between the same pairs of callers we obtained a graph with $n = 2,100,000$ nodes and $m = 48,400,000$ directed edges that include 10,800,000 bidirectional pairs.

Bidirectional edges are crucial in some of our applications since they show mutual connection compared to a one-directional call to e.g. a public service number. When considering bidirectional edges only, approximately 30,000 users (1.5%) become isolated from the giant component of the large graph.

Information on the name and geographic location of the settlement (city, village, suburban district) of the nodes is used in several of our experiments and models. Settlement sizes (Fig. 3.2, right) follow a distribution very close to lognormal with the exception of a very heavy tail of Hungary's capital Budapest of near 600,000 users. In a rare number of cases the data consists of subpart names of settlements resulting in a relatively large number of settlements with one or two telephone numbers; since the total number of such nodes is negligible in the graph, we omit cleaning the data in this respect.

	large	small
Number of nodes (thousands)	2,072	74
Nodes outside giant component	130	15
Time coverage (months)	8	12
Number of edges, directed (thousands)	48,400	3,965
Number of edges, bidirectional (thousands)	10,800	706

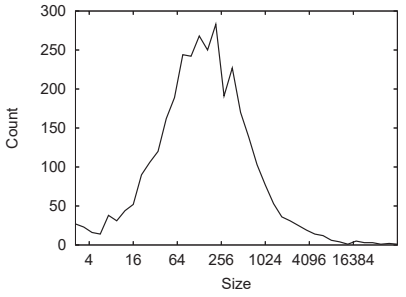


Figure 3.2: **Left:** Main parameters of the call graphs. **Right:** Distribution of the user number by settlements in the data; the capital Budapest of near 600,000 users is trimmed.

The graph has strong top-down regional structure with large cities appearing as single clusters. These small world power law graphs are centered around very large degree nodes and very hard to split. In most parameter settings of the original spectral method we are left with a large cluster of size near that of the Budapest telephone users.

3.1.3 UK2007-WEBSPAM host graph

The third data set is the host graph of the UK2007-WEBSPAM crawl of Boldi et al. [29] that contains 111,149 hosts and 1,836,441 directed weighted edges. The hosts are labeled with the top level Open Directory [127] categories as in [85]. The list of the largest categories are seen in Fig. 4.13, right.

Web content categorization is a research area that abounds with opportunities for practical solutions. The performance of most traditional machine learning methods is limited by their disregard for the interconnection structure between web data instances (nodes). At the same time, relational machine learning methods often do not scale to web-sized data sets. Beyond the choice of the method, creative feature generation and selection can greatly improve web categorization performance. For instance, it has been shown [11] that in addition to textual contents, the link structure of web hosts offer valuable clues in predicting their type.

3.1.4 Cluster quality measures

Next we define the graph and sociodemographic based quality measures we use for evaluating the output of a clustering algorithm. Let there be N users with N_k of them in cluster C_k for

$k = 1, \ldots, m$. The *cluster ratio* is the number of edges between different clusters divided by $\sum_{i \neq j} N_i \cdot N_j$. The *weighted cluster ratio* is obtained by dividing the total weight of edges between different clusters by $\sum_{i \neq j} w_{ij} N_i \cdot N_j$, where w_{ij} is the total weight of the edges between cluster i and j.

Modularity, a measure known to suit social networks well [161] is defined as follows:

$$Q = \sum_{\text{clusters } s} \left[\frac{d(C_s, C_s)}{M} - \left(\frac{d(C_s, \overline{C_s})}{2M} \right)^2 \right], \tag{3.1.1}$$

where M is the total weight of the edges and $d(X, Y)$ is the weight of the edges with tail in X and head in Y. Since modularity is not balanced by the cluster size, we use *normalized network modularity* [149], defined as

$$Q_{norm} = \sum_{\text{clusters } s} \frac{N}{N_k} \left[\left(\frac{d(C_s, \overline{C_s})}{2M} \right)^2 - \frac{d(C_s, C_s)}{M} \right],$$

We remark that the authors in [149] negate nomalized modularity compared to modularity; we stick to their notation and use negative values of normalized modularity. In our experiments normalized modularity turned out to be instable and we suspect it may not be an appropriate measure for cluster quality.

Telephone users as nodes have rich sociodemographic attributes beyond graph theory. We may measure clustering quality by the entropy and purity of the geographic location or other external property within the cluster. By using the notation of the previous subsection, let $N_{i,k}$ denote the cluster confusion matrix, i.e. the number of elements in cluster k from settlement i and let $p_{i,k} = N_{i,k}/N_k$ denote the ratio within the cluster. Then the *entropy* E and *purity* P are defined as

$$E = \frac{-1}{\log m} \sum_k \frac{N_k}{N} \sum_i p_{i,k} \log p_{i,k} \quad \text{and} \quad P = \frac{1}{N} \sum_k \max_i N_{i,k},$$

where the former is the average entropy of the distribution of settlements within the cluster and the latter measures the ratio of the "best fit" within each cluster.

3.2 Link prediction

In this section we provide a link prediction experiment that, given an observed period of usage, predicts pairs of users (edges, or links) that will appear in a future time period. By such an experiment we may investigate node similarity measures for finding important connections and ignoring "accidental" unimportant ones.

Link prediction is to our best knowledge first investigated by Liben-Nowell and Kleinberg [111] with very similar motivations in an earlier paper of Newman [123]. They consider a wide variety of methods based on the neighborhood and the ensemble of paths corresponding to the pair of nodes in question. Next we describe a selection of these methods including those that performed best in their experiments. Then we will show link prediction measurements over the telephone call graph.

Several algorithms were designed to evaluate node-to-node similarities in networks that can be used to give alternate, similarity based weights to node pairs. We refer to [111] for an exhaustive list of the available methods ranging from co-citation to more complex measures such as max-flow/min-cut-based similarities defined in [112]. These weights are used in applications outside the link prediction area: [77, and many more] apply them to improve clustering quality; co-citation is for example first used in [82] as an elementary step of trust propagation.

Similarity in a telephone call graph is best characterized by the undirected graph since communication is typically bidirectional regardless of the actual caller–callee direction. We also have a choice to use cost, duration or number of calls as a weight of a pair of users, or we may ignore weights and consider an unweighted graph. We will compare both the input directed graph, its transpose by changing the direction of each edge, or the undirected version arising as the union of the previous two graphs. We will refer to the three variants as *directed*, *reversed* and *undirected* versions. For an edge weight function $d : V \times V \to \mathbf{R}$ we use $d^-(u, v) = d(v, u)$ for the reversed and $d^\leftrightarrow = d + d^-$ for the undirected version. We extend this notion for an arbitrary similarity measure $\mathrm{sim}(u, v)$ computed over edge weights d and compute $\mathrm{sim}^-(u, v)$ over d^- and $\mathrm{sim}^\leftrightarrow(u, v)$ over $d^\leftrightarrow$.

In the discussion below we identify scalability as the main challenge for computing node similarities. Computing all pairs' similarities is computationally challenging even for our networks of a few million nodes since the entire quadratic size similarity matrix would occupy several Terabytes. Notice that the experiments of Liben-Nowell and Kleinberg [111] were conducted on much smaller data. As one possibility we may calculate similarity only for existing edges. The resulting scheme downweights unimportant edges but is unable to add "uncaught contacts" to the network. As a possible solution to finding the potential strong relationship between pairs of nodes not connected by an edge, we may find all pairs with weight above a given threshold by fingerprinting techniques; these techniques will however be specific to the given similarity measure.

3.2.1 Neighborhood based methods

The first broad class of measures for the strength of the connection between two nodes u and v depends on the strength of the overlap between the neighborhood of u and v. Next we define the measures of cocitation, Jaccard, Adamic/Adar and cosine similarities.

The *cocitation* or *common neighbors* $\text{coc}(u, v)$ is defined as the number of common in-neighbors of u and v. This measure turned out most effective for Web spam classification [23]. By the notation of edge directions, $\text{coc}^{-}(u, v)$ denotes the bibliographic coupling (nodes pointed to by both u and v) and $\text{coc}^{\leftrightarrow}(u, v)$ is the total number of (undirected) common neighbors. We may also define a variant of cocitation that is downweighted by degree as $\text{coc}(u, v)/d(u) \cdot d(v)$.

The Jaccard coefficient $\text{Jac}(u, v)$ is the ratio of common neighbors within all neighbors. If we let $\Gamma(u)$ denote the neighbors of u, then

$$\text{Jac}(u, v) = |\Gamma(u) \cap \Gamma(v)|/|\Gamma(u) \cup \Gamma(v)|$$

For a weighted graph we may divide the total weight of edges leading to common neighbors by the total weight of edges from u and v. Unfortunately this measure does not correlate the pairs of weights ux and vx for common neighbors x. Due to this problem we observe a particularly poor performance in the case when we have a single strongly related neighbor x of u and y of v and the Jaccard similarity is 0. If edges uy and vx receive an "accidental" low weight, the Jaccard coefficient however immediately becomes very high while the actual similarity remains very low.

Cosine similarity fixes the above problem of the Jaccard coefficient. We consider the row of the adjacency matrix corresponding to node u as vector $\overline{u}$. The cosine similarity of nodes u and v is simply $\cos(u, v) = \overline{u}^T\overline{v}$. We may similarly define $\cos^{-}(u, v)$ over the transpose matrix and $\cos^{\leftrightarrow}(u, v)$ over the sum.

Adamic and Adar [3] define a measure that downweights high degree common neighbors as they may occur simply by chance. The Adamic/Adar measure is defined as

$$\text{AdamicAdar}(u, v) \sum_{z \in \Gamma(u) \cap \Gamma(v)} 1/\log|\Gamma(z)|.$$

Simple neighborhood based edge weighting schemes already pose computational challenges for large networks since filling the quadratic size similarity matrix is infeasible. Next, we describe the min-hash fingerprint of Broder et al. [36] to identify all pairs with weight above a given threshold. Based on the min-hash fingerprint and embedding, more complex approximation of related measures such as cosine is described in [41].

The fingerprint of node u under a random permutation[2] π of all nodes is defined as the minimum neighbor of u in the ordering of π:

$$\text{fingerprint}_{\pi}(u) = \min\{\pi(u') : u' \in \Gamma(u)\}.$$

Where $\pi(u)$ denotes the position of u in the permutation. For two nodes u and v the fingerprints

[2]In fact π does not have to be random: the weaker so-called min-wise independence requirement suffices.

coincide if and only if the minimum of $\Gamma(u) \cup \Gamma(v)$ under π belongs to $\Gamma(u) \cap \Gamma(v)$, hence the probability of this event is equal to the Jaccard similarity of the nodes. By generating a sufficiently large number of fingerprints (in practice 100–10000) we may approximate $\mathrm{Jac}(u, v)$ as the fraction of the fingerprints of u and v that coincide.

3.2.2 Multi-step propagation: Methods based on path ensembles

Advanced node similarity measurement methods are capable of using a part or all of the entire path ensemble connecting the given pair of nodes and not just the neighborhood that corresponds to length 2 paths. In this section we briefly introduce such methods and the efficient algorithms [69, 143] for approximately computing them.

Path ensemble measures became widespread with the success story of Google's PageRank [34, 128] and other hyperlink-based quality measures [32, 98]. Since its introduction in 1998, PageRank remains the prominent example of a path ensemble measure as it is defined as a certain multi-step generalization of the degree defined below. In fact, PageRank is best known as a quality measure based on the recursive reasoning that the importance of a node is high if it is pointed to by several important nodes. Personalized PageRank, a variant of PageRank dating back to the original paper of Page et al. [128], is however capable of measuring the strength of the connection between a node or a weighted set of nodes and another node.

Next we introduce notation for (personalized) PageRank. Let A denote the stochastic matrix corresponding to the random walk on the network, i.e.

$$A_{ij} = \begin{cases} 1/\mathrm{outdeg}(i) & \text{if host } i \text{ points to } j, \\ 0 & \text{otherwise.} \end{cases}$$

The *PageRank* vector $p = (p_1, \ldots, p_N)$ is defined as the solution of the following equation [34]:

$$p_r = (1 - c) \cdot \sum_{v=1}^{N} p_v A_{vu} + c \cdot r \ , \tag{3.2.1}$$

where $r = (r_1, \ldots, r_N)$ is the teleportation distribution and c is the teleportation probability with a typical value of $c \approx 0.15$. We get the PageRank if we set all r_i to $1/N$; for general r we get PageRank personalized on r. If $r = \chi(w)$ consisting of all 0 except for node w where $\chi_w(w) = 1$, then we personalize on the single vertex and let PPR_w denote the corresponding vector.

As we will see, variants of the PageRank of u personalized on v define similarity measures of u and v. These values are however even more expensive to compute for all u, v than the neighborhood based measures of the previous subsection. Below we describe two reformula-

tions of the PageRank equation that yield scalable approximation algorithms for several related measures. The Monte Carlo simulation procedure of Fogaras and Rácz [70] is a general method to estimate random walk based path ensemble measures. The Dynamic Programming algorithm [91] gives rise to approximation algorithms [143] that can also be used for estimating several weighted neighborhood values similar to those of [20].

The first PageRank reformulation was noticed independently by [68, 91]. The (personalized) PageRank of a vertex is equal to the probability of a random walk terminating at the given vertex where the length is given by a geometric distribution: we terminate at step t with probability $c \cdot (1-c)^t$. To justify this, notice that PPR_w (and PageRank in general) can be rewritten as a power series

$$\text{PPR}_w = \chi(w) \cdot \sum_{t=0}^{\infty} c(1-c)^t \cdot A^t. \tag{3.2.2}$$

The term $\chi(w)A^t$ corresponds to a random walk of length t starting at w and $c \cdot (1-c)^t$ is the probability of termination. The above equation also explains why $\text{PPR}_w(u)$ is a path ensemble based similarity of u and w: we enumerate all paths from w to u by giving exponentially decreasing weight to long paths.

As described by Fogaras and Rácz [70], equation (3.2.2) can be approximated by randomly generating paths with length according to the geometric distribution $c(1-c)^t$. They empirically observe that 1000 samples suffice for a good quality approximation even in large graphs. For algorithmic details and error analysis we refer to [70].

A second, equivalent reformulation of the path summing formula (3.2.2) is the Decomposition Theorem proved by Jeh and Widom [91] stating that a node's personalized PageRank vector is expressible with the average personalized PageRank vector of its out-neighbors giving extra weight to the node itself:

$$\text{PPR}_u = c\chi_u + (1-c) \cdot \sum_{v:(uv)\in E} \text{PPR}_v / d^+(u). \tag{3.2.3}$$

As observed by Sarlós et al. [143], the above equation is the right choice for computing all PPR values above certain threshold. The other altenative is the path summation formula (3.2.2); however there we accumulate all error when entering a large in-degree node and hence we must compute partial results fairly exact. The dynamic programming equation (3.2.3) in contrast averages all partial results into a new PPR_u and because of averaging we do not amplify error at large in-degrees. In particular we may safely discard all partial $\text{PPR}_u(v_i)$ values below threshold for further computations since the total error will remain below the threshold in (3.2.3).

Given the path summation reformulation (3.2.2) of personalized PageRank we may define several variants of weighting neighbors at distance k. We may define reachability and exact reachability by $d^k(u,v)_{\text{reach}} = 1$ if v is reachable from u by a walk over k edges, 0 otherwise,

respectively $d^k_{\text{exact}}(u,v) = 1$ if v is reachable from u in exactly k steps and over no shorter paths, 0 otherwise. We may use the number and the weighted number of such walks in the definition: $d^k_{\text{num}}(u,v)$ is the number of walks over k edges that reach from u to v and $d^k_{\text{wnum}}(u,v)$ is the probability of reaching v when starting at u and at each step choosing a random neighbor with probability proportional to the outgoing edge weights.

One example of the generalization of path summation is the historically earliest path ensemble measure of Katz [95] dating back to the fifties defined as

$$\text{Katz}_\beta(u,v) = \sum_{t=1}^{\infty} \beta^t \cdot d^k_{\text{num}}(u,v);$$

weighted Katz measure arises if we replace d^k_{num} by d^k_{wnum}. These measures can be approximated by both of the above methods.

More complex path ensemble measures arise as the multi-step variants of cocitation and the Jaccard coefficient. Jeh and Widom [90] define SimRank as a multi-step generalization of downweighted cocitation as follows:

$$\text{Sim}^{(k)}(u_1,u_2) = \begin{cases} (1-c) \cdot \sum_{v_1 \in \Gamma(u_1), v_2 \in \Gamma(u_2)} \text{Sim}^{(k-1)}(v_1,v_2)/(d^-(u_1)d^-(u_2)) & \text{if } u_1 \neq u_2 \\ 1 & \text{if } u_1 = u_2. \end{cases}$$

In an alternative formulation [143] SimRank equals the total weight of pairs of walks

$$\begin{aligned} v_1 = w_0, w_1, \ldots, w_{k-1}, w_k &= u \\ v_2 = w'_0, w'_1, \ldots, w'_{k-1}, w'_k &= u \end{aligned}$$

that both end at u and one of them comes from v_1 while the other one from v_2. The weight of the pair of walks is the *expected* $(1-c)$ *meeting distance* as defined in [90]:

$$(1-c)^k/(d^-(w_1)\cdots d^-(w_k) \cdot d^-(w'_1)\cdots d^-(w'_k)); \tag{3.2.4}$$

notice that we get cocitation back for $k = 1$. Fogaras and Rácz [69] describe XJaccard as the weighted sum of Jaccard coefficients of the distance k neighborhoods as follows:

$$\text{XJac}(u,v) = \sum (1-c)^k \text{Jac}^{(k)}(u,v)$$

where $\text{Jac}^{(k)}(u,v)$ is the Jaccard similarity of the distance k neighborhood of u and v. These measures can be approximated in a similar way of PageRank by path sampling in equation (3.2.4) [69]; more complex space optimal algorithms are also described in [143] for certain SimRank variants.

3.2.3 Link prediction experiments

In order to illustrate the strength of the methods from the previous two subsections, we set up the following link prediction experiment. We compute the similarity measures based on the first four months (training period) of the large data (Fig. 3.2). We use these similarity measures as a prediction for the next four months period (test period). Given a threshold value, we measure precision and recall as

$$\begin{aligned} \text{Precision} &= \frac{|\{\text{edges above threshold}\} \cap \{\text{actual edges at months 5–8}\}|}{|\{\text{edges above threshold}\}|}; \\ \text{Recall} &= \frac{|\{\text{edges above threshold}\} \cap \{\text{actual edges at months 5–8}\}|}{|\{\text{actual edges at months 5–8}\}|}. \end{aligned}$$

We also introduce weighted recall to bias quality measures towards correctly identifying heavy weight edges and penalizing less for low weight ones. By letting w_e denote the weight of an edge in the second (test) four months period we let

$$\text{WRecall} = \frac{\sum\{w_e : e \text{ has weight above threshold}\}}{\sum_e w_e}.$$

Results of the link prediction experiment are shown in Fig. 3.3 in terms of precision–recall (top) and precision–weighted recall (bottom) curves. The curves are obtained by varying the threshold. Weighted recall is significantly higher in all cases, indicating that heavy weight edges are easier to predict; the relative order of the quality of the predictions however remains the same for both curves. Similarly to the findings of Liben-Nowell and Kleinberg [111], variants of Katz performed best among path ensemble measures while cosine among neighborhood measures. Due to the limitations of the visualization, we could not place all variants in Fig. 3.3. We only show Jaccard and Adamic/Adar in addition to the above neighborhood based measures as well as preferential attachment, a trivial baseline method defined by the product of the degrees of the two node in question.

A key difference in our experiment compared to [111] is that we predict all edges in the test period, not just new edges. This distinction is visible when comparing the left and right graphs of Fig. 3.3. The left hand side graphs show lower quality because those u–v similarity measures do not take into account whether u and v are connected by an edge or not. The measures on the right hand side count the existence of an edge between u and v either as a part of the Katz measure, or else directly add it to neighborhood measures (common neighbors, cosine, Jaccard). Best performance is obtained when the logarithm of the edge weight (in time duration) is added; these measures are shown in Fig. 3.3, right.

Since in our task we also have to predict edges that already existed in the training period, aggregated time duration of an edge turns out to be a very strong predictor in itself. This

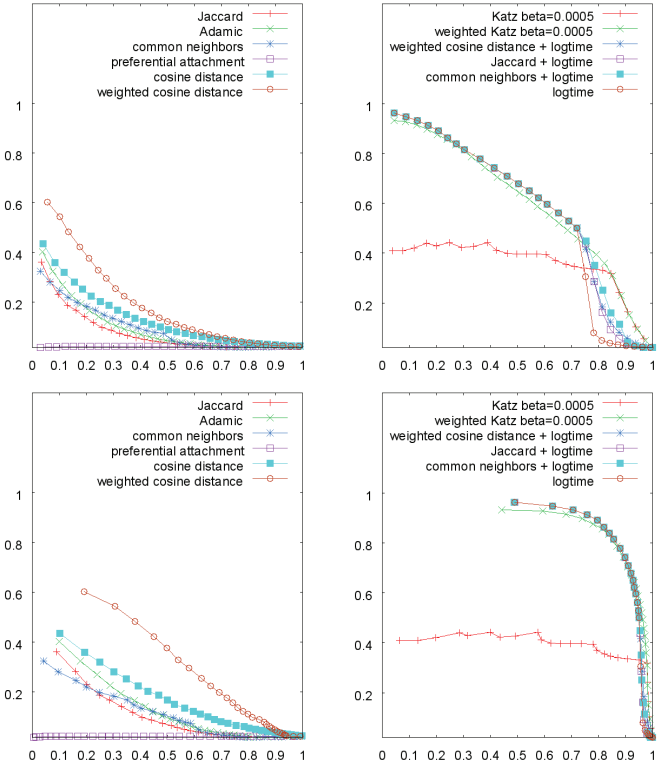


Figure 3.3: Precision-recall curves of the link prediction experiment trained on the first 4 months and tested on the last 4 months of the large graph. **Top graphs:** precision–recall curves. **Bottom graphs:** precision and *weighted* recall curves. Precision is over the vertical axis in both cases. **Left graphs:** curves corresponding to measures that exclude the (logarithmic) weight of the training period. **Right graphs:** curves for measures that all include a logarithmic edge weight term.

measure is outperformed only for high recall ranges when neighborhood based measures are capable of identifying additional new edges in the network. In this range Katz performs very well.

We also draw attention to the importance of edge weights. In the graphs we use weights in two places. Time duration values are on one hand entries in the vectors that define the weighted cosine measure; on the other hand they modify the path probabilities in weighted Katz. Weighted cosine turns out to be the best neighborhood based measure while weighted Katz is only defeated by logarithmic weight plus weighted cosine at certain ranges of recall.

To draw a conclusion, we have surveyed a number of node similarity measures based on both neighborhood overlap and entire path ensembles and sketched some scalable algorithms and techniques to approximate them. Having analyzed precision-recall curves of a link prediction experiment, we have observed best performance for weighted cosine and Katz similar to the findings of [111]. In the next sections we will also use these measures as alternative weights to the edges as an input to further processing.

3.3 Network topology: navigation in the Small World

The so-called "small world" phenomenon was first observed in social networks by Milgram [118] who examined the average path length for social networks of people in the United States and found an average distance of six steps, later referred to as "six degrees of separation". Social and other networks exhibit low diameter as demonstrated by several results. As an example, in [7] the diameter of the World Wide Web is measured. As another one, in [44] the low diameter of a wide class of networks obeying degree distribution constraints such as power law distribution is proved. Telephone call networks fall into this category; in this section we observe low distances and efficient navigation in our graphs.

The first graph model explaining the small world phenomenon is described by Watts and Strogatz [156] and later extended by Kleinberg [96, 97] who also incorporate a path finder algorithm in the model that uses local information only in each step. Notice that Milgram's experiment [118] did not only show low average distances but also a capability of network nodes to find these paths based solely on their local information on the network. In fact information is not entirely local, we must have at least certain global information on the target node. It is easy to see that if all we can determine whether a given node is the target or not, we have no choice other than to perform a random walk until we reach the target, an algorithm that visits each node several times on average. In Milgram's experiment among others the geographic location of the target was given to the node. By using this information, an algorithm may for example select the current node's neighbor geographically closest to the target, a clear advantage over a random walk.

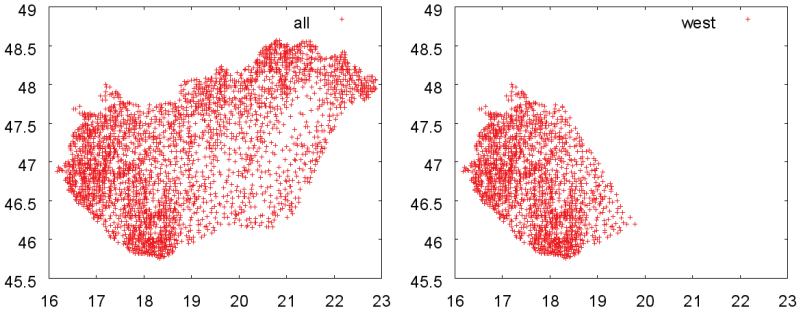


Figure 3.4: **Left:** The geographical distribution of the customer network within Hungary with axes showing latitude and longitude. **Right:** the western region avoiding the capital Budapest at latitude 47° 28' 19" N and longitude 19° 03' 01" E selected for the experiment.

Kleinberg's celebrated small world model [97] describes the following network with a geographic path finder algorithm. We obtain a small world graph in a d dimensional attribute space by placing nodes in a d-dimensional grid, connecting all pairs within a constant distance and adding long range contacts with probability proportional to r^{-d} where r is the distance. In a recent extension, Kumar et al. [101] observe over the network of bloggers that nodes are not distributed evenly over the grid and r^{-d} should be replaced by $t(r)^{-1}$ where $t(r)$ is the number of users of distance at most r.

In the following we show measurements for the distribution of the distance between pairs of contacts and fit the results to Kleinberg's model. We conduct this experiment over the graph with nodes formed by the users in the large data of Fig. 3.2; in Fig. 3.4 we show the geographic location of these users. Since the capital Budapest locates roughly 1/3 of all nodes that are hence of geographic distance 0, we have selected the western region of near 750,000 users shown in the right of Fig. 3.4. By Kleinberg's model the distribution of the distance of a given node from its neighbors is inverse polynomial; for a two-dimensional area as in Fig. 3.4 the exponent is -2. Our measurements shown in Fig. 3.5 justify this model as follows. On the left hand side we see a noisy behavior due to large cities and in particular the capital; the distribution fits slightly better to r^{-1} but the quality is poor. The quality of the fit however significantly improves if we remove the effect of the capital: the western region with no city containing 50,000 or more users fits r^{-2} very well.

We also describe a path finder experiment where for each intermediate node we greedily select the neighbor geographically closest to the target node. Unfortunately we have no information other than location, hence we say that the path terminates if it reaches the settlement of the target. In Fig. 3.6 we show the number of steps required by this greedy routing algorithm to

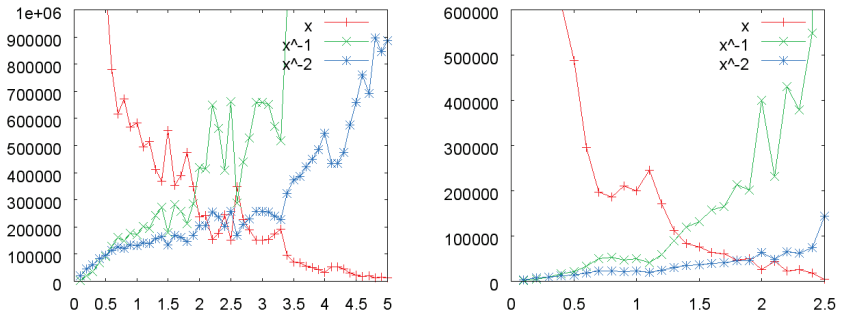


Figure 3.5: **Left:** The number of edges as a function of the distance between its endvertices. **Right:** The same measurement over the western region (Fig. 3.4) to filter out the effect of the capital. Both figures contain the data transformed to test linear fit with functions x^{-1} and x^{-2}. Horizontal axes show distance in the coordinates of Fig. 3.4.

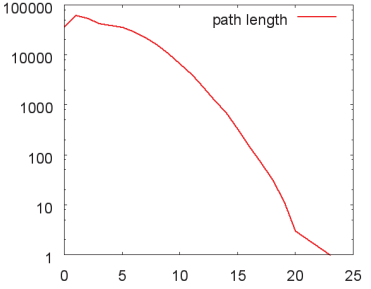


Figure 3.6: The distribution of the greedy distance routing steps for a random pair of 1M users.

find the target for a set of 1,000,000 randomly selected pairs of users. We see a fast exponential decay in the number of paths required beyond distance 10. The distances we measure are close to the "magical" six in Milgram's experiment despite of the fact that our network is much smaller and we are satisfied with simply reaching the settlement location of the node. Notice however, that we are unable to use information other than location in intermediate steps and thus for example we never move to a node within the same settlement. In a practical scenario, in contrast, a participant may know a neighbor who have relatives near the target area and the walk may advance very close after a local step that is seemingly useless in the model. In this sense the participants in Milgram's experiment were able to use much richer data for their routing decision.

An external, non-graph-theoretic attribute can explain social contacts if edges are more sim-

ilar in this attribute than non-edges. However as social networks are small world, short distances in a low dimensional attribute space may only occur if we allow long range contacts as well.

3.4 Conclusion and bibliographic notes

While most of the results in this Chapter are introductory, some experiments appeared in [24] where I conducted most of the experiments described in this Chapter.

We have surveyed some results of social network modeling and analysis with illustrations over the call logs of major Hungarian telephone companies with millions of users, long time range, and sufficiently strong sociodemographic information on the users. We have analyzed the results of link prediction and route finding and compared the performance of various node similarity measures in these tasks.

Chapter 4

Spectral Partitioning of Social Networks

In this chapter we describe and compare several clustering algorithms on the real life networks of Section 3.1. Unlike in the examples of [92], in our graphs the "right" clustering is by no means obvious but, similar to the findings of [92], the goodness measures can be fooled. The typical examples of practically useless spectral splits have uneven sizes or disconnected clusters; in certain cases the clustering procedure simply wastes computational resources for unnecessary steps, a phenomenon reported in particular for power law graphs [109]. We believe our findings are beyond "it works well on my data" and apply to a more general class of social networks or other small-world power law graphs.

Prior to our work, spectral clustering was known to fail for large power law graphs with several partly successful attempts [109]. As a particular example, previously the only known large scale formation of the LiveJournal blogger network was the Russian user group [81, 165]. By our methods we reveal clusters arranged by location, age and certain types of interest such as religion.

When clustering large social networks, spectral methods tend to chop off tentacles attached loosely to a densely connected larger subset, resulting in a disconnected part and keeping the dense component in one [109]. While even the optimum cluster ratio cut might have this structure, the disconnected cluster consists of small graph pieces that each belong strongly to certain different areas within the dense component. In addition a disconnected graph has multiple top eigenvalues, meaning that we must compute eigenvectors separately for each connected component. However if we treat each connected component as a separate cluster, we obtain an undesired very uneven distribution of cluster sizes.

As a related area, the HITS [98] ranking algorithm is a direct application of the SVD since the hub and authority ranks correspond to the first left and right singular vectors. It has been known for long that HITS is instable [124] and it should be applied for subgraphs only. We believe that the reason is the same as for the failure of spectral partitioning. In particular by using our preprocessing method we avoid the Tightly Knit Community (TKC) phenomenon

caused by communities that are small on a global level but still grab the first (or, as we show, even the first many) principal axes. Lempel et al. [110] are the probably the first who identify the TKC problem in the HITS algorithm, their algorithmic solution (SALSA) however turns out to merely compute in- and out-degrees [32]

In our main results we define a set of heuristics that prevent low level communities from overtaking the first principal axes. Our method is based on the combination of the removal of TKCs [110] and the contraction of long tentacles. We build on the recent findings of Xu et al. [161] who identify bridges across TKCs as the main reason for the failure of graph partitioning methods.

As a core procedure, we modify the SCAN method of Xu et al. [161] by using a different similarity measure that can then be approximated to yield an efficient implementation for very large graphs. We remark the importance of the choice of SCAN for dense community removal; other methods such as those of [52, 56] either produce huge communities or add too few nodes into communities to have effect on graph partitioning.

While the method of Xu et al. [161] promises a partitioning of the network based on the identification of community cores, similar to several other core finder methods [65, 66] the cores identified are of small size on the global scale and cannot yield information on the global structure. Xu et al. [161] test their method in part on real graphs that are mere few 100 nodes and in part on graphs generated for particular use for their algorithm based on the construction of [122] that first determines target clusters and then connects nodes within the same cluster with higher probability than between two clusters but otherwise independent at random.

Even though in our observations community core finder algorithms are insufficient in themselves for partitioning very large networks, these methods however can be used prior to spectral partitioning to remove a large number of cores that act as TKCs by attracting a large number of principal vectors. When combining core removal and tentacle contraction, we obtain high level distinctive characteristics of the connections between network that include geography, religion and age. Of particular interest are our findings on how LiveJournal friends are organized on the top level of the network along these dimensions.

While a comprehensive comparison of clustering algorithms is beyond the scope of this work, we justify the use of a top-down hierarchical clustering by observing that telephone call graphs and social networks in general are small world power law graphs. Small world implies very fast growth of neighborhood that strongly overlap; power law implies high degree nodes that locally connect a large number of neighboring nodes. Recent bottom-up alternatives such as clique percolation [52] suffer from these phenomena: the extreme large number of small (size 5 or 6) cliques do not only pose computational challenges but also connect most of the graph into a single cluster; the number of larger sized cliques however quickly decays and by using them we leave most of the nodes isolated or in very small clusters. The superiority of spectral

clustering over density based methods is also suggested in [42] for document collections.

Practical evaluation of spectral clustering in graphs is investigated mainly in the area of netlist partitioning [10] with the recent exception of the findings of Lang [108, 109]. He suggests semidefinite programming techniques to avoid imbalanced cuts, however the reported running times are several hours for a single cut even for 10 million edge graphs. Techniques to scale the semidefinite programming based approaches and a comparison of the performance remains future work.

While implementation issues of SVD computation are beyond the scope of this Chapter, we compare the performance of the Lanczos and block Lanczos code of `svdpack` [26] and our implementation of a power iteration algorithm. Hagen et al. [86] suggest fast Lanczos-type methods as robust basis for computing heuristic ratio cuts; others [43, 92] use power iteration. Since the SVD algorithm itself has no effect on the surrounding clustering procedure, we only compare performances later in Section 4.3.1. Our key findings on implementing spectral clustering in real-world networks are:

- We give a k-way hierarchical clustering algorithm variant that outperforms the recently described Divide-and-Merge algorithm of Cheng et al. [43] both for speed and accuracy.
- We give two pre-processing heuristics that remove community cores and loosely connected tentacles to successfully attack the hardest-to-partition real networks.
- Compared to the Laplacian $D - A$ typically used for graph partitioning, we show superior performance of the normalized Laplacian $D^{-1/2}AD^{-1/2}$ introduced for spectral bisection in [147] and [54] as the relaxation of the so-called normalized cut and min-max cut problems, respectively. We are aware of no earlier systematic experimental comparison. While in [53, 147] both described, their performance is not compared in practice; Weiss [158] reports "unless the matrix is normalized [...] it is nearly impossible to extract segmentation information" but no performance measures are given; finally [155] give theoretic evidence for the superiority of normalization.
- We compare various edge weighting schemes, in particular introduce a neighborhood Jaccard similarity weight. This weight outperforms the best logarithmic weighting in certain cases, justifying the discussion of [92, Section 2] that the input matrix should reflect the similarity of the nodes instead of their distance.
- We introduce size balancing heuristics that improve both the geographic homogeneity and the size distribution of the clusters formed by the algorithm. These methods outperform and completely replace Lin-Kernighan type heuristics proposed by [54].
- We partially justify previous suggestions to use several eigenvectors [10, 113]; however we observe no need for too many of them.

4.1 Two recent clustering methods

Before turning to our spectral clustering experiments we describe two recent community detection algorithms and show that although they are reported to work well for small networks, they are unable to identify homogeneous large scale structures in our telephone call networks.

In this section we give a brief sketch of the clique percolation algorithm of Derényi and Palla et al. [52, 130]. Instead of presenting the deep theory behind the method, our emphasis is on the implementational details for very large graphs. Note that in [130] measurements over only a few thousand node graphs are described. In another closely related publication on clique percolation [129] a telephone call graph with over 4 million users is used as input and the time evolution of the clusters is analyzed. In that paper however no details are given on how to implement the algorithm for such a large scale problem, the communities analyzed have less than 1000 users, and no global analysis of the community size distribution is given. In order to easier relate all results in this Chapter we hence show some additional measurements on clique percolation in this subsection.

Clique percolation grows communities from k-cliques as building blocks. A k-clique is a complete subgraph over k nodes; we call two such cliques *connected* if they share $k-1$ nodes. Clique percolation identifies the maximal connected components of k-cliques as possibly overlapping clusters.

We describe a possible implementation of the clique percolation algorithm we use that is based on the modification of the APRIORI frequent itemset mining algorithm [5]. In one step described in Algorithm 4.1.1 we enumerate all $(k+1)$-cliques for increasing values of $k = 0$, $1, \ldots$ and store them as lexicographically ordered sets in a trie. Given a $(k+1)$ element set $U \cup u \cup v$ with $|U| = k-1$ and u and v of index higher than all elements in U, this set is a $(k+1)$-clique if and only if $U \cup u$ and $U \cup v$ are k-cliques and there is an edge between u and v. This condition can easily checked since $U \cup u$ and $U \cup v$ are both children of a level $k-1$ element U of the trie of k-cliques.

Algorithm 4.1.1 Construction of the trie O of $(k+1)$-cliques from the trie I of k-cliques.

```
O ← empty trie
for all sets U in the trie I on level k − 1 do
  for all pairs of nodes u and v such that U ∪ u and U ∪ v ∈ I do
    if u and v are connected by an edge then
      add U ∪ u ∪ v to O
return O
```

Clique percolation in our experiment suffers from two problems due to the scale of the input data. Firstly for dense graphs there are simply too many cliques to enumerate: in the top table of Fig. 4.1 we see that, when run on the small graph (Fig. 3.2), we have to discard the largest degree nodes. In this mere 74,000 node graph there are 39M 3-cliques without filtering; for

	small, x nodes discarded				large
x	500	2000	9000	24000	
edges	3180989	2409073	1212284	400892	26758776
3-cliques	14747294	6669967	1400158	171657	11194867
4-cliques	37100792	10197104	803202	34512	2914247
5-cliques	72445357	12113978	354230	5251	688114
6-cliques	n/a	12171488	138053	687	225204
7-cliques	n/a	11122318	48277	52	105853
8-cliques	n/a	9625892	14743	2	48860

k	5	4	3
number of cliques	688114	2914247	11194867
largest cluster	13988	88785	1605756
number of components	74841	323077	360745
nodes in at least one component	358544	1105731	1798855

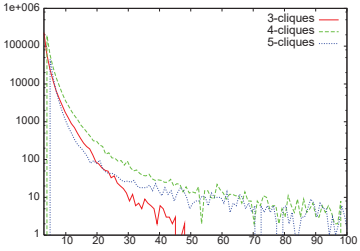


Figure 4.1: **Top:** The size of the cliques for the small and large graphs of Fig. 3.2. For the small graph the highest degree x nodes are discarded for different values of x. **Bottom:** properties of clusters over the large graph with $k = 3$, 4 and 5 and the histogram of their size distribution.

4-cliques we ran out of memory at over 100M enumerated.

Secondly, it turns out to be hard to balance between an extreme large number of small cliques that do not only pose computational challenges but also connect most of the graph into a single cluster and a low number of larger cliques that leave most of the nodes isolated or in very small clusters. As seen in Fig. 4.1, bottom, in our large graph for $k = 5$ most of the nodes remain in isolation while for $k = 3$ we are left with a giant component. The best $k = 4$ choice places slightly more than half of the nodes in some cluster, although many of these clusters are very small as seen in Fig. 4.1, bottom left.

As another proposed solution, the assumption of Xu et al. [161] is that there exist hub vertices in a network that connect or, in their terminology, bridge many clusters. Therefore they define the SCAN algorithm that selects pairs of vertices with a concentration of common neighbors as candidate intra-cluster nodes limited by parameter ϵ. Hubs, as opposed to intra-cluster nodes, are then characterized by the distraction of neighbors. Finally cores are formed by nodes that have at least μ neighbors within the core.

The key step in the SCAN algorithm is the selection of edges between pairs of nodes whose neighborhood similarity is above a threshold ϵ. In the original algorithm of Xu et al. [161], with $\Gamma(u)$ denoting the neighbors of u, the similarity is measured as

$$\sigma(u, v) = |\Gamma(u) \cap \Gamma(v)|/\sqrt{|\Gamma(u)||\Gamma(v)|}.$$

For power law graphs, in particular for the Web graph in our experiments, however the running time for computing $\sigma(u, v)$ is very large due to the huge neighborhood sets $\Gamma(u)$ involved. Hence, we use the Jaccard similarity

$$\text{Jac}(u, v) = |\Gamma(u) \cap \Gamma(v)|/|\Gamma(u) \cup \Gamma(v)|$$

that we approximate by 100 min-hash fingerprints [35] as described in Section 3.2.1.

The SCAN Algorithm 4.1.2 (modified to use Jaccard similarity) proceeds as follows. First it discards edges that connect pairs of dissimilar nodes below threshold ϵ; these edges may bridge different dense regions [161]. Then nodes with more than μ remaining edges are considered as community cores. Finally, connected components of cores along remaining edges are augmented by neighboring non-core nodes. The resulting components $\mathcal{C}$ may overlap at these augmented vertices; these vertices are called *hubs* in [161] since they provide connectivity across different communities.

Tests with various parameter settings of the SCAN algorithm over the large graph (Fig. 3.2) are shown in Fig. 4.2. As an overall evaluation we observe that SCAN is unable to detect communities of size beyond a few tens in our network. The better the parameter setting we use, the more communities are found in the size range around 20 nodes. In more detail we compare

Algorithm 4.1.2 Modified SCAN.

input: ϵ: similarity threshold of neighbors within same core, μ: size threshold of neighborhood within core

output: list of communities

for all edges uv **do**
  compute approximate $\text{Jac}(u, v)$ by min-hash fingerprints
$E' \leftarrow \{(uv) : \sigma(u, v) \geq \epsilon\}$
$V' \leftarrow \{u : \deg_{E'}(u) \geq \mu\}$
compute the connected components $\mathcal{C}$ of V' with edges E'
for all components C of $\mathcal{C}$ **do**
  Add all vertices to C that are connected to C by edges of E'
return $\mathcal{C}$

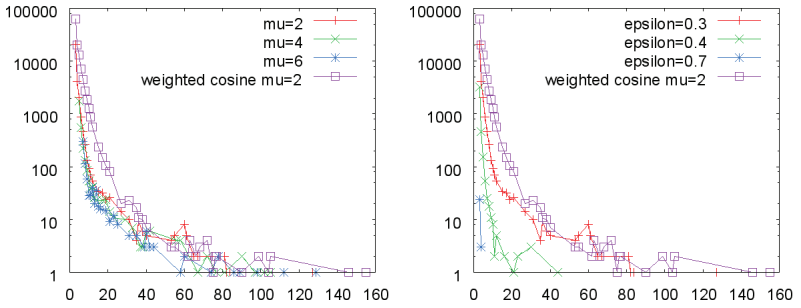


Figure 4.2: The distribution of the community sizes identified by the SCAN algorithm. **Left:** the effect of varying μ for a low $\epsilon = .3$ similarity value. **Right:** the effect of varying ϵ for the weakest $\mu = 2$ core size bound. Both figures contain the weighted cosine input graph with the best parameter settings in addition to Jaccard as in Algorithm 4.1.2.

the Jaccard similarity as described in Algorithm 4.1.2 and the weighted cosine similarity; due to computational constraints the latter is only computed for the existing edges of the network. Even in this weaker setting weighted cosine identifies more meaningful communities. Best performance is obtained by the smallest possible $\mu = 2$ and small ϵ. If we increase either μ (Fig. 4.2, left) or ϵ (Fig. 4.2, right), the number of communities identified decreases at all size ranges.

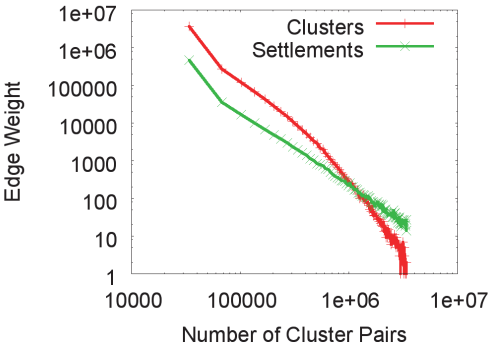


Figure 4.3: Distribution of the edge weights across different clusters, for a spectral clustering and a trivial clustering obtained by considering one settlement as one cluster. The vertical axis contains the total edge weight in seconds and the horizontal axis shows the number of cluster pairs with the given weight between them.

4.2 Algorithms

4.2.1 Spectral clustering: an experiment

In order to illustrate the importance of the choice of parameters and input graph weights for clustering as well as the issues on the quality measurement of the outcome, we present spectral clustering experiments over the *large* graph of over 2M nodes and 10M bidirectional edges (for detailed data see Fig. 3.2).

One may argue whether clustering reveals additional information compared to the settlements themselves as "ground truth" clusters. We give a positive answer to this question by showing that the distribution of the total call duration across different clusters is optimal for those obtained by spectral clustering. In Fig. 4.3 we form two graphs, one with a node for each settlement and another with a node for each (spectral) cluster. The weight of an edge between two such nodes is the total call duration between the corresponding clusters. We observe that the edge weights follow a power law distribution in both graphs. The graph obtained by spectral clustering has a much smaller exponent and the edges across clusters have much smaller weight, indicating an improved arrangement of weight mass for the spectral clusters. In fact we use settlement information as an external validation tool for our experiments and not as ground truth.

We have several choices to extract the network based on telephone calls between users: we may or may not ignore the direction of the edges and weight edges by number of calls, duration or price, the latter emphasizing long range contacts. First of all we may try to use the total cost

or duration directly as a weight in the adjacency matrix. It turns out that the Lanczos algorithm converges extremely slowly. While it converges within a maximum of 120 iterations in all other cases, 900 iterations did not suffice for a single singular vector computation with raw values. We therefore use $1+\log w_{ij}$ where w_{ij} is either the total cost or duration between a pair of users i and j.

We also investigate a Jaccard and a cosine similarity based weight of user pairs in line with the remark of Kannan et al. [92] who suggest node similarities for input weights. These methods yield weights between 0 and 1, and clustering in the reweighted graph has quality similar to the logarithm of call cost or duration. For both similarity measures we use $1 + \text{sim}_{ij}$ to distinguish non-edges from low-weight edges.

4.2.2 Small cluster redistribution heuristics

The key in using spectral clustering for power law graphs is our small cluster redistribution heuristics described in the next subsection. After computing a k-way split, we test the resulting partition for small clusters. First we try to redistribute nodes to make each component connected. This procedure may reduce the number of clusters. In a degenerate case we may even be left with a single cluster; in this case the output is rejected and clustering fails.

Algorithm 4.2.1 `redistribute`$(C_1, \ldots, C_k)$: Small cluster redistribution

for all C_i **do**
  $C_i' \leftarrow$ largest connected component of C_i
  if $|C_i'| <$ `limit` $\cdot\, |C_1 \cup \ldots \cup C_k|$ **then**
    $C_i' \leftarrow \emptyset$
Outlier $= (C_1 - C_1') \cup \ldots \cup (C_k - C_k')$
for all $v \in$ Outlier **do**
  $p(v) \leftarrow j$ with largest total edge weight $d(v, C_j')$
for all $v \in$ Outlier **do**
  Move v to new cluster $C_{p(v)}$
return all nonempty C_i

We give a subroutine to reject very uneven splits that is used in both our Divide-and-Merge implementation (Section 4.2.2) and in k-way clustering (Section 4.2.3). Given a split of a cluster (that may be the entire graph) into at least two clusters $C_1 \cup \ldots \cup C_k$, we first form the connected components of each C_i and select the largest C_i'. We consider vertices in $C_i - C_i'$ outliers. In addition we impose a relative threshold `limit` and consider the entire C_i outlier if C_i' is below limit.

Next we redistribute outliers and check if the resulting clustering is sensible. In one step we schedule a single vertex v to component C_j with $d(v, C_j)$ maximum where $d(A, B)$ denotes the number of edges with one end in A and another in B. Scheduled vertices are moved into their

clusters at the end so that the output is independent of the order vertices v are processed. By this procedure we may be left with less than k components; we will have to reject clustering if we are left with the entire input as a single cluster. In this case we either try splitting it with modified parameters or completely give up forming subclusters.

4.2.3 K-way hierarchical clustering

Algorithm 4.2.2 k-way hierarchical clustering

while we have less than `cnum` clusters **do**
  $A \leftarrow$ adjacency matrix of largest cluster C_0
  Project $D^{-1/2}AD^{-1/2}$ into first d eigenvectors
  For each node i form vector $v_i' \in R^d$ of the projection
  $v_i \leftarrow v_i'/||v_i'||$
  $(C_1, \ldots, C_k) \leftarrow$output of k-means$(v_1, \ldots, v_{|C_0|})$
  Call `redistribute`$(C_1, \ldots, C_k)$
  Discard C_0 if C_0 remains a single cluster

In our benchmark implementation we give k, the number of subclusters formed in each step, d, the dimension of the SVD projection and `cnum`, the required number of clusters as input. Algorithm 4.2.2 then always attempts to split the largest available cluster into $k' \leq k$ pieces by k-means after a projection onto d dimensions. Note that k-means may produce less than the prescribed number of clusters k; this scenario typically implies the hardness of clustering the graph. If, after calling small cluster redistribution (Algorithm 4.2.1), we are left with a single cluster, we discard C_0 and do not attempt to split it further.

In our real life application we start out with low values of d and increase it for another try with C_0 whenever splitting a cluster C_0 fails. We may in this case also decrease the balance constraint.

Notice the row normalization step $v_i \leftarrow v_i'/||v_i'||$; this step improves clustering qualities for our problem. We also implemented column normalization, its effect is however negligible.

4.2.4 Divide-and-Merge Baseline

The Divide-and-Merge algorithm of Cheng et al. [43] is a two phase algorithm. In the first phase we recursively bisect the graph: we perform a linear scan in the second eigenvector of the Laplacian sorted by value to find the optimal bisection. The algorithm produces $\texttt{cnum}_0$ clusters that are in the second phase merged to a required smaller number `cnum` of clusters by optimizing cut measures via dynamic programming.

In order to adapt the Divide-and-Merge algorithm originally designed for document clustering [43], we modify both phases. First we describe a cluster balancing heuristic based on

Algorithm 4.2.3 Divide and Merge: Divide Phase

while we have less than $\texttt{cnum}_0$ clusters **do**
  $A \leftarrow$ adjacency matrix of largest cluster C_0
  Compute the second largest eigenvector v' of $D^{-1/2}AD^{-1/2}$
  Let $v = D^{-1/2}v'$ and sort v
  $i \leftarrow \texttt{ratio\_init}$
  while C_0 is not discarded **do**
    Find $1/i \leq t \leq 1 - 1/i$ such that the cut

$$(S, T) = (\{1, \ldots, t \cdot n\}, \{t \cdot n + 1, \ldots, n\})$$

    minimizes the cluster ratio
    $(C_1, \ldots, C_\ell) \leftarrow \texttt{redistribute}(S, T)$
    if $\ell > 1$ **then**
      Discard C_0 and add clusters $C_1, \ldots, C_\ell$
    else
      if $i = 3$ **then**
        Discard cluster C_0
      else
        $i \leftarrow i - 1$

Algorithm 4.2.1 for the divide phase. Then for the merge phase we give an algorithm that produces low cluster ratio cuts, a measure defined below in this section. In [43] the merge phase of the divide-and-merge algorithm is not implemented for cluster ratio. Since this measure is not monotonic over subclusters, we give a new heuristic dynamic programming procedure below.

We observed tiny clusters appear very frequent in the Divide phase (Algorithm 4.2.3) as described in Section 4.2.2. Splits along the second eigenvector are apparently prone to find a disconnected small side consisting of outliers. In this case the small component heuristics of Algorithm 4.2.1 are insufficient themselves since we are starting out with two clusters; if we completely redistribute one, then we are left with the component unsplit. We hence introduce an additional balancing step with the intent to find connected balanced splits along the second eigenvector. We could restrict linear scan to an 1/3-2/3 split; in many cases this however leads to a low quality cut. Hence first we weaken the restriction to find an $1/\texttt{ratio\_init}$–$(1 - 1/\texttt{ratio\_init})$ cut and gradually decrease the denominator down to 3. We stop with the first cut not rejected by Algorithm 4.2.1. If no such exists, we keep the cluster in one and proceed with the remaining largest one.

Now we turn to the Merge phase (Algorithm 4.2.4). Our goal is to optimize the final output for cluster ratio defined below. Let there be N users with N_k of them in cluster k for $k = 1, \ldots, m$. The *cluster ratio* is the number of calls between different clusters divided by $\sum_{i \neq j} N_i \cdot N_j$. The *weighted cluster ratio* is obtained by dividing the total weight of edges between different clusters by $\sum_{i \neq j} w_{ij} N_i \cdot N_j$ where w_{ij} is the total weight of edges between

Algorithm 4.2.4 Merge Phase

for all clusters C_0 from leaves up to the root **do**
  if C_0 is leaf **then**
    $\text{OPT}_\text{n}(C_0, 1) = 0, \text{OPT}_\text{d}(C_0, 1) = |C_0|$
  else
    Let $C_1, \ldots, C_\ell$ be the children of C_0
    for i between 1 and total below C_0 **do**
      $\text{numer}(i_1, \ldots, i_\ell) \leftarrow 0; \text{denom}(i_1, \ldots, i_\ell) \leftarrow 1$
      for all $i_1 + \ldots + i_\ell = i$ **do**
        $\text{numer}(i_1, \ldots, i_\ell) \leftarrow \sum_{j \neq j'} d(C_j, C_{j'}) + \sum_{j=1\ldots\ell} \text{OPT}_\text{n}(C_j, i_j)$
        $\text{denom}(i_1, \ldots, i_\ell) \leftarrow \sum_{j \neq j'} |C_j| \cdot |C_{j'}| + \sum_{j=1\ldots\ell} \text{OPT}_\text{d}(C_j, i_j)$
        if $\dfrac{\text{OPT}_n(C_0, i)}{\text{OPT}_d(C_0, i)} > \dfrac{\text{numer}(i_1, \ldots, i_\ell)}{\text{denom}(i_1, \ldots, i_\ell)}$ **then**
          $\text{OPT}_n(C_0, i) = \text{numer}(i_1, \ldots, i_\ell); \quad \text{OPT}_d(C_0, i) = \text{denum}(i_1, \ldots, i_\ell)$

clusters i and j.

In order to compute the optimal merging upwards from leaves by dynamic programming (Algorithm 4.2.4) we aim to use an idea similar to computing cluster ratio when linearly scanning in the Divide step as described in Section [42]. Unfortunately however cluster ratio is not monotonic in the cluster ratio within a subcomponent; instead we have to add the numerator and denominator expressions separately within the subcomponents. We can only give a heuristic solution below to solve this problem.

In order to find a good cluster ratio split into i subsets of a given cluster C_0, we try all possible $i_1 + \ldots + i_\ell = i$ split sizes within subclusters $C_1, \ldots, C_\ell$. By the dynamic programming principle we assume good splits into i_j pieces are known for each subcluster C_j; as we will see, these may not be optimal though. For these splits we require the values $\text{OPT}_n(C_j, i_j)$ and $\text{OPT}_d(C_j, i_j)$, which denote the optimal numerator and denomiator values of the i_j-way split of cluster C_j. If we use the corresponding splits for all j, we obtain a split of cluster ratio

$$\frac{\sum_{j \neq j'} d(C_j, C_{j'}) + \sum_{j=1\ldots\ell} \text{OPT}_\text{n}(C_j, i_j)}{\sum_{j \neq j'} |C_j| \cdot |C_{j'}| + \sum_{j=1\ldots\ell} \text{OPT}_\text{d}(C_j, i_j)}$$

for the union of the subcomponents. Note however that this expression is not monotonic in the cluster ratio of subcomponent j, $\text{OPT}_n(C_j, i_j)/\text{OPT}_d(C_j, i_j)$, and the minimization of the above expression cannot be done by dynamic programming. As a heuristic solution, in Algorithm 4.2.4 we always use the optimal splits from children. Even in this setting the algorithm is inefficient for branching factor more than two; while in theory Merge could be used after k-way partitioning as well, the running time is exponential in k since we have to try all (or at least most) splits of i into $i_1 + \ldots + i_\ell$.

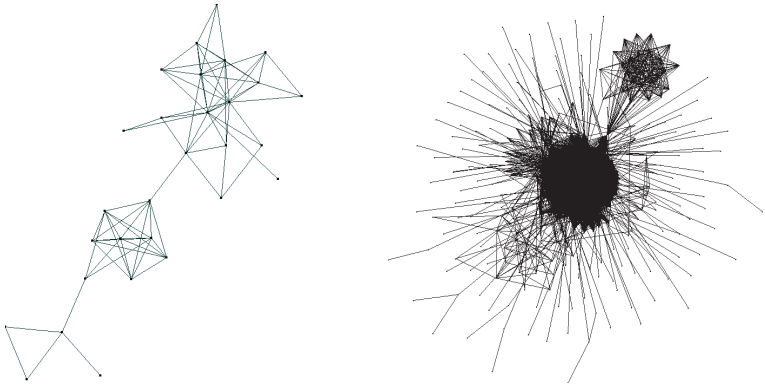


Figure 4.4: Left: A 82-node subgraph of the LiveJournal Friends network, with two cores and several short tentacles. Right: a similar 317-node subgraph of the UK2007-WEBSPAM host graph.

4.2.5 Tentacles: loosely connected regions

Algorithm 4.2.5 Tentacle contraction.

input: $d_{\max}$: maximum degree of a tentacle node.
output: graph G' with all tentacles contracted.

while node v of degree $\leq d_{\max}$ exists in G' **do**
  Contract v to its neighbor u with lowest degree in G'
  Record $v \to u$ for tentacle set reconstruction

We introduce a pre-processing heuristics for handling loosely connected parts of the network. We recursively contract all nodes that have degree below a threshold into a neighbor. In this way tentacles are eliminated and close communities are moved in proximity of each other.

In a recursive definition we say that a node belongs to a *tentacle* if its degree is not more than a prescribed value $d_{\max}$; we use $d_{\max} = 3$. As long as there are tentacle nodes in the graph, we contract them into (one of) their neighbors with smallest degree. In this way we may create new small degree nodes; the procedure may recursively continue. By recording the contractions we may also reconstruct all nodes that get contracted into a final node; such a set of nodes is called a tentacle. The procedure is described in Algorithm 4.2.5. We note that the definition of a tentacle depends on the order of contractions and hence we only use it as a preprocessing heuristic and do not use tentacles for characterizing a particular node.

4.2.6 Tightly knit communities and the SCAN algorithm

Another main ingredient of our algorithm consists of the removal of community cores seen in Fig. 4.4 or, in another terminology, tightly knit communities (TKC) before singular value decomposition. Several authors observe difficulties caused by the TKCs: Lempel and Moran [110] investigate hyperlink based ranking on the Web; very recently [161] identifies hubs that bridge between several TKCs as the main difficulty in network partitioning.

Several algorithms are proposed to identify community cores. Flake et al. use network flows [65] or min-cut trees [66]; Xu et al. [161] use an agglomerative method that prefers core nodes and avoids bridges that connect more than one TKC. All these methods however suffer from the abundance of very small communities with no superimposed larger scale structure that network flow based heuristics could exploit.

Our heuristic solution is based on the Structural Clustering Algorithm for Networks (SCAN) algorithm [161]; however instead of using moderate parameters to build large clusters directly as community cores, we use SCAN with restrictive values and remove 1–5% of the nodes that belong to TKC prior to PCA.

Algorithm 4.2.6 Spectral Clustering.

input: k: desired branching factor of the cluster hierarchy.
output: hierarchical clustering

while desired number or cluster size is not reached **do**
  Select largest cluster C_0 and induced subgraph G
  $Q_1, \ldots, Q_s \leftarrow$ cores given by SCAN(G, ϵ, μ)
  $G' \leftarrow G - \bigcup Q_i$
  $G'' \leftarrow$ Contract all tentacles in G'
  $A \leftarrow$ adjacency matrix of G''
  Project $D^{-1/2}AD^{-1/2}$ into first d eigenvectors
  For each node i form vector $v_i' \in R^d$ of the projection
  $v_i \leftarrow v_i'/||v_i'||$
  $(C_1, \ldots, C_k) \leftarrow$output of k-means$(v_1, \ldots, v_{|C_0|})$
  Call `redistribute`$(C_1, \ldots, C_k, Q_1, \ldots, Q_s)$
  Discard C_0 if C_0 remains a single cluster
return all discarded and remaining clusters

4.2.7 Components of the algorithm

First the pre-filtering heuristics (Sections 4.2.5 and 4.2.6) are applicable in general to obtain globally meaningful principal axes. These heuristics not only improve the clustering quality by filtering out the globally relevant network structure, but they also decrease running times.

Then we apply one of the two graph bisection relaxation methods (SVD or SDP). For the SVD we may choose to use the Laplacian or the weighted Laplacian matrix. These methods

project the graph into a d-dimensional vector space [40]. We use k-means to get an initial split into more than two parts as suggested first by [166], or a much simpler approach for the $d = 1$ case as described in 4.2.3.

For SVD we use the Lanczos code of `svdpack` [26] and for SDP we use Burer and Monteiro's solver [38].

After computing a 2-way or k-way split we test the resulting partition for small clusters. First we try to redistribute nodes to make each component connected Algorithm (4.2.2). This procedure may reduce the number of clusters; when we are left with a single cluster, the output is rejected.

Finally, to get a hierarchical clustering algorithm we can choose from two algorithms. The first algorithm in Section 4.2.3 is based on k-way hierarchical clustering as described among others by Alpert et al. [10]; the second one in Section 4.2.4 on the more recent Divide-and-Merge algorithm [43].

Both of the algorithms target at balancing the output clusters. The original Divide-and-Merge algorithm of [43] achieves this simply by producing more clusters than requested and merging them in a second phase. We observed this algorithm itself is insufficient for clustering power law graphs since for our data it chops off small pieces in one divide step. In a recursive use for hierarchical clustering the number of SVD calls hence becomes quadratic in the input size even if only a relative small number of clusters is requested.

The two main ingredients of our algorithm consist of the removal of small dense regions and contracting long interconnecting tentacles. In Fig. 4.4 we see typical subgraphs of the entire network that consist of several small community cores, two of which are seen, with low degree nodes loosely connected to some of them or interconnecting pairs of them. Since PCA is unable to select from the abundance of small cores, it falls into the trap of the so-called TKC effect [110] by selecting the most dominant such structure that is still very small on the scale of the entire network. We demonstrate that after the proposed preprocessing these traps are avoided and meaningful principal axes are found.

4.3 Experiments

In this section we describe our experiments performed mainly on the Hungarian Telecom call detail record and, in order to extend the scope of applicability, on the UK2007-WEBSPAM crawl and the LiveJournal blogger friends network.

The experiments were carried out on a cluster of 64-bit 3GHz P-D processors with 4GB RAM each. Depending on algorithms and parameter settings, the running time for the construction of 3000 clusters is in the order of magnitude of several hours or a day for the Hungarian Telecom data, the largest of the graphs used in our experiments.

Algorithm	$d = 2$	$d = 5$	$d = 10$	$d = 15$	$d = 20$	$d = 25$
Lanczos	17	24	44	47	55	96
Block Lanczos	19	34	66	105	146	195
Power	15	40	95	144	191	240

Table 4.1: Running times for the d dimensional SVD computation by various algorithms, in minutes.

4.3.1 Telephone graph

Evaluation of Singular Value Decomposition algorithms

In our implementation we used the Lanczos code of `svdpack` [26] and compared it with block Lanczos and a power iteration developed from scratch. While block Lanczos runs much slower, it produces the exact same output as Lanczos; in contrast power iteration used by several results [43, 92] is slightly faster for computing the Fiedler vector but much less accurate; computing more than two dimensions turned out useless due to the numerical instability of the orthogonal projection step. Improving numerical stability is beyond the scope of this work and we aware of no standard power iteration implementation. Running times for the first split are shown in Table 4.1; in comparison the semidefinite programming bisection code of Lang [109] ran 120 minutes for a much smaller subgraph ($n = 65,000$, $m = 1,360,000$) while for the entire graph it did not terminate in two days.

We remark that modifications of `svdpack` are necessary to handle the size of our input. After removing the obsolete condition on the maximum size of an input matrix, we abstracted data access within the implementation to computing the product of a vector with either the input matrix or its transpose.

The entire running time for producing 3000 clusters depend more on the parameter settings than the choice of Divide-and-Merge vs. k-way partitioning. All runs took several hours up to a day; only the slowest Divide-and-Merge with `limit` $= 100$ runs over a day. Since the number of possible parameters is very large, we omitted running time graphs.

Divide-and-Merge vs. k-way hierarchical algorithm

The comparison of various input matrices to both divide-and-merge and k-way hierarchical clustering is shown in Fig. 4.5. Most importantly we notice the weighted Laplacian $D^{-1/2}AD^{-1/2}$ significantly outperforms the unweighted $D - A$ in all respects. Call length and call cost behave similar; as expected, the former yields geographically more homogeneous clusters by underemphasizing long distance calls, while the latter performing better for the cluster ratio measure. The logarithm of the price or duration performs very close to Jaccard reweighting with no clear winner.

When comparing Divide-and-Merge and k-way partitioning (Fig. 4.5) we observe the supe-

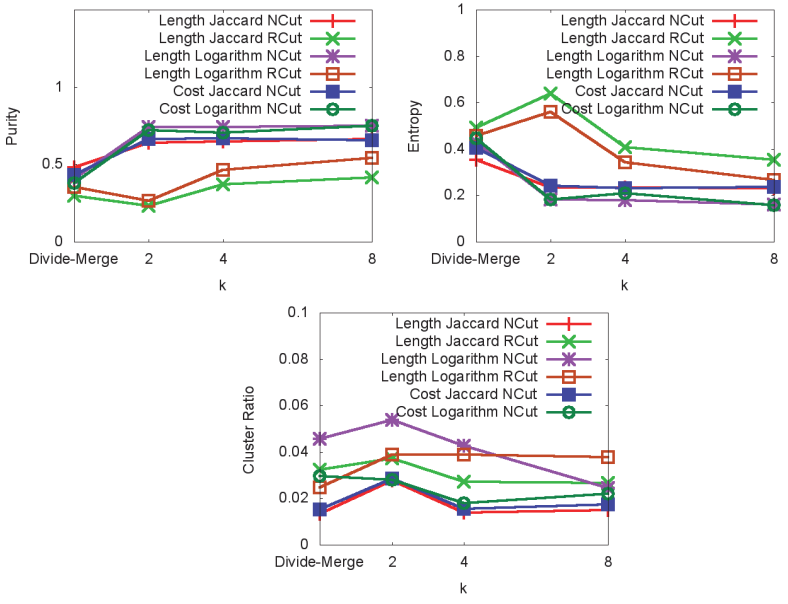


Figure 4.5: Evaluation of various reweighting techniques over the adjacency matrix for purity (left), entropy (right) and cluster ratio (bottom) of the arising clusters on the vertical axis. Curves correspond to combinations of unweighted vs. weighted Laplacian (NCut for normalized cut relaxation, as opposed to RCut, ratio cut relaxation), length vs. cost based, and Jaccard vs. logarithmic weight input matrices. Four different algorithms, Divide-and-Merge bipartition as well as k-way partition with $d = 30$ for three values $k = 2$, 4 and 8 are on the horizontal axis.

riority of the latter for larger k. For $k = 2$ we basically perform Divide without Merge; the poor performance is hence no surprise. For $k = 4$ however the small cluster redistribution heuristic already reaches and even outperforms the flexibility of the Merge phase in rearranging bad splits.

Size limits and implications on the size distribution of clusters

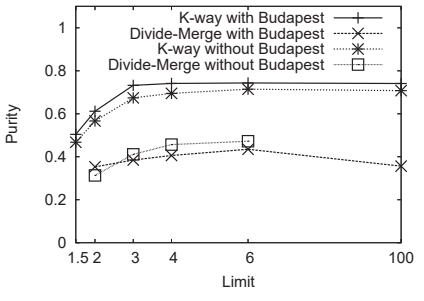
In Fig. 4.6 we see the effect of changing `limit` for the k-way and Divide-and-Merge algorithms. Recall that in Algorithm 4.2.1 used as subroutine in both cases, the parameter `limit` bounds the ratio of the smallest cut from the average. If this is very large (100 in the Figure), we are left with no constraint. If however it is close to one, we enforce very strict balancing that deteriorates clustering quality. The optimal values lie around $4 \ldots 6$; these values are also optimal for running time. Very large values, in particular for Divide-and-Merge, slow algorithms down by only marginally reducing the largest cluster size after the costly SVD computation.

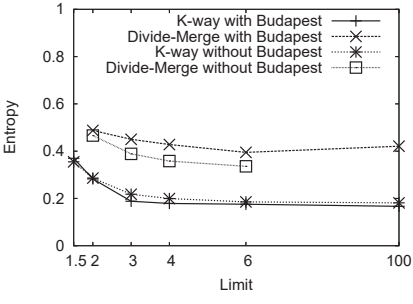
We checked that the strict balancing constraints required for efficiency has no negative effect on cluster quality. This is clearly seen for purity and entropy in Fig. 4.6, top. Notice however the unexpected increase of cluster ratio (middle left) for large values; this is due to the fact that the densely connected near 600,000 Budapest users could only be split with liberal balancing conditions as also seen in the table of largest remaining cluster sizes in Fig. 4.6, middle right. While splitting Budapest has no effect on purity or entropy, it adds a large number of edges cut in cluster ratio. For this reason we repeated the experiment by removing Budapest users to see no negative effect of the strict balance constraint on the clustering quality measures anymore. We did not include normalized modularity in the figures since, as we will see in Section 4.3.2, normalized modularity turned out to be instable.

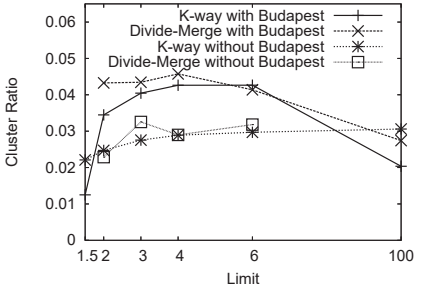
Notice the superiority of the k-way algorithm over Divide-and-Merge is also clear for their best parameter settings of Fig. 4.6, top.

We also remark here that we implemented a Lin-Kernighan type point redistribution at cut borders proposed by [54] but it had negligible effect on the quality.

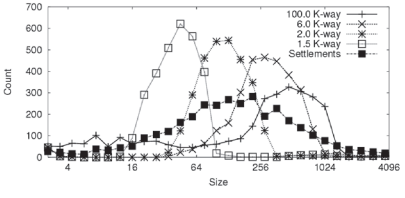
Besides clustering quality, we also look at how "natural" are the cluster sizes produced by the algorithms in Fig. 4.6, bottom. We observe strong maximum cluster size thresholds for Divide-and-Merge: that algorithm forces splitting hard regions for the price of producing a negatively skewed distribution of a large number of small clusters that are of little practical use. With the exception of Divide-and-Merge with no limits we never split Budapest users as seen from the top list (Fig. 4.6, middle right). When repeating the experiment by discarding Budapest users, the huge clusters disappear.







Algorithm &Limit	1.	2.	1. w/o Budapest
K-way 1.5	947571	134450	161515
K-way 2.0	828543	104421	86904
K-way 3.0	746869	63433	73683
K-way 4.0	746869	49240	61922
K-way 6.0	746869	55287	61922
K-way 100.0	712557	49577	60667
D-M 2.0	956415	643520	195418
D-M 3.0	817094	120746	163169
D-M 4.0	817094	120746	163169
D-M 6.0	725058	190964	150666
D-M 100.0	357618	297407	timeout



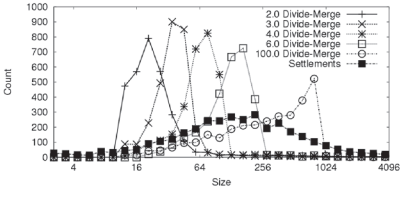


Figure 4.6: Effect of size limits on clustering quality, $k = 4$ and $d = 30$ for purity, entropy (top) and cluster ratio (middle left). The size of the largest and second largest remaining cluster as well as the largest one after the removal of Budapest lines (middle right). Distribution of cluster sizes for k-way hierarchical partitioning (k-way) and Divide-and-Merge (Divide-Merge) for various parameters of the size limit (bottom).

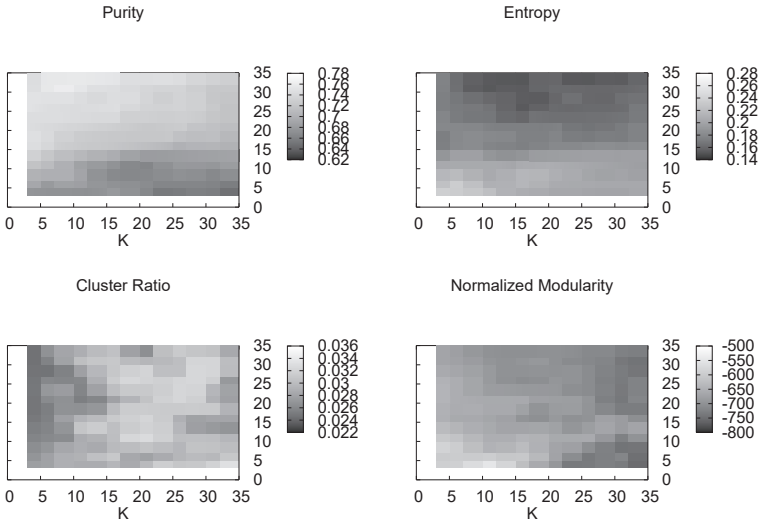
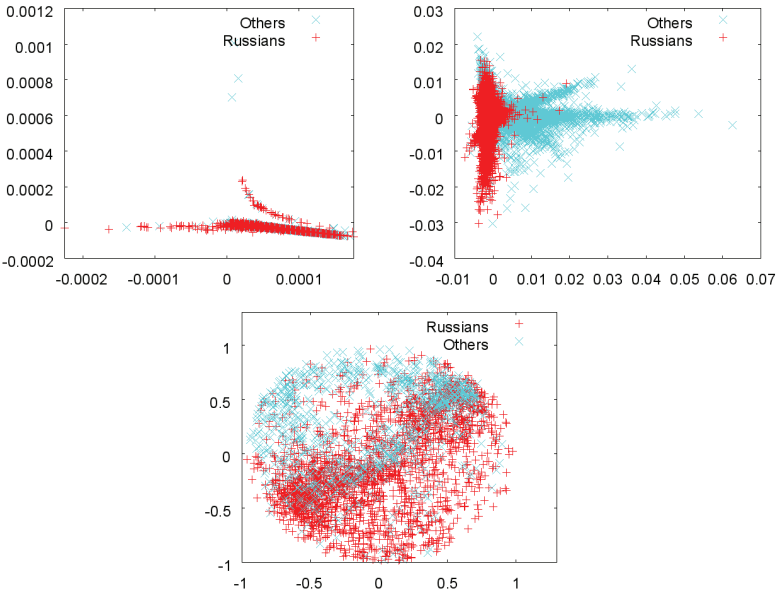


Figure 4.7: Relation between dimensions d (vertical), branching k (horizontal) and quality (darkness) for purity (top left), entropy (top right), cluster ratio (bottom left) and normalized modularity (bottom right). The darker the region, the better the clustering quality except for purity where large values denote good output quality.



	Purity (location)	Entropy	Norm. Mod.
No heuristics	0.501557	1.182126	3.556151
Tentacle + SCAN	0.504375	0.634847	11.781545

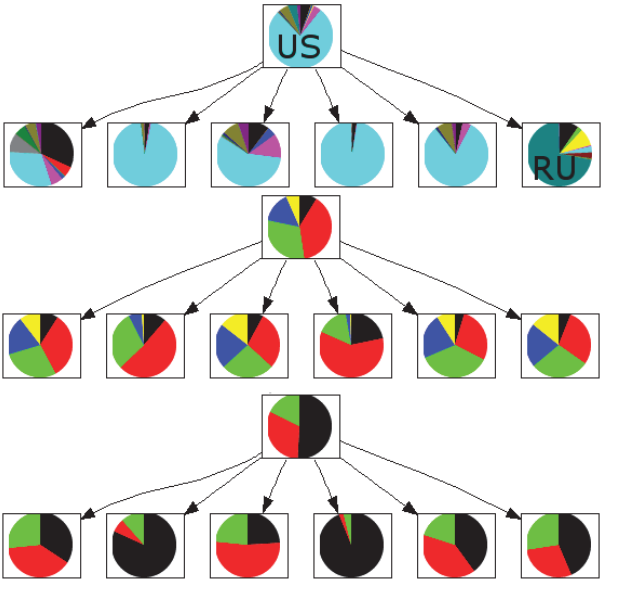
Figure 4.8: Principal axes 4 and 5 within the Russian cluster before (top) and after (middle) the removal of cores and tentacles as well as two dimensions of the SDP relaxation (bottom). The quality of subpartitioning the first cluster is in the bottom table.

4.3.2 The LiveJournal Friends Network

The effect of more dimensions

Our first observation is that direct spectral partitioning of the non-Russian LiveJournal is impossible due to a singular value sequence with even the 100th largest above 0.99. In accordance, the principal axes are non-characteristic. For example in Fig. 4.8 we observe no difference between Russians and other nations within the Russian cluster (UA, BY, EE etc.) without preprocessing; this distinction becomes however strongly visible by using our algorithm. In Table 5.1 we see that 15 dimensions suffice for a balanced partitioning only if all preprocessing heuristics are applied.

The running times and cluster quality measures are summarized in Table 4.3. The most



No	Size	characteristics
1	60 681	International/Europe
2	699 199	US college
3	447 694	International/English speaking + SG, PN
4	170 257	US high school
5	729 604	US Atheist/Pagan
6	216 832	Russian

Figure 4.9: Partition of LiveJournal users into six, with the distribution of location (top), age (middle) and religious interest (bottom). Characteristics of the parts 1–6 (left to right) are shown in the table.

729093	music	26266	Jesus
480443	movies	17828	Paganism
353412	reading	9845	Theology
331027	writing	9752	Atheism
312060	friends	6834	Democrats
251376	art	5054	Jesus Christ
229519	photography	2486	Republicans
217465	books	1677	Democrat
214479	dancing	1291	Republican

Table 4.2: Number of users who express certain type of interest. Left: the top list. Right: polarized interest categories.

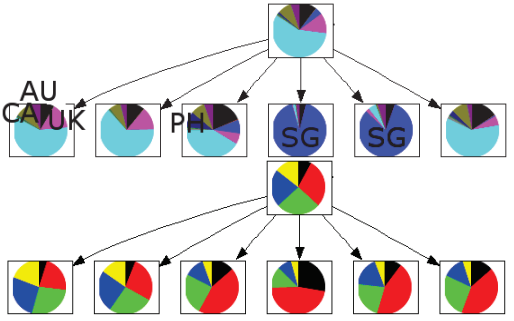


Figure 4.10: Subpartition of LiveJournal cluster No. 3, with the distribution of location (top) and age (bottom). Dominant location US is distributed into three clusters (#1, 2 and 6) with an age distribution moving from older groups towards a majority 16–18 from subcluster 1 to 6.

important observation is the huge running time difference between SDP and SVD with only minor differences in cluster quality. As for the reliability of the measures, since entropy and purity relate the clustering to a ground truth, we may take them more reliable measures as the pure graph based ones. In this sense cluster ratio apparently does not form a reliable comparison probably due to its dependence on the number of clusters. The remaining three measures, although noisy in certain cases, do not show major differences in judgment.

The quality improvement for clustering is justified by observing that our method makes PCA easier while not destroying the essential network properties. When comparing the quality of the partitions given by different algorithms, we observe that SVD with the heuristics works always nearly as good as SDP; in fact for LiveJournal, the hardest instance it outperforms SDP in entropy and normalized modularity. The very low normalized modularity of SDP here may indicate an unfortunate split; note that the purity values are very close to .8, the fraction of US location that corresponds to a random split. Here our heuristics greatly improve SDP as well; for the other data sets however SDP performs in general better without them.

The advantages of our method become even clearer when we dig deeper into subclusters. We considered a component (No. 1 in Fig. 4.9) with 60,681 nodes and 228,644 edges where partitioning is not even possible without our heuristics. The cluster quality parameters are seen in the table on figure 4.8.

Countries and regions

Characteristic countries are those that, in a 6-level hierarchical clustering into 3000 components, constitute 20% of at least one cluster. Other than US that constitutes near 80% of the users (see

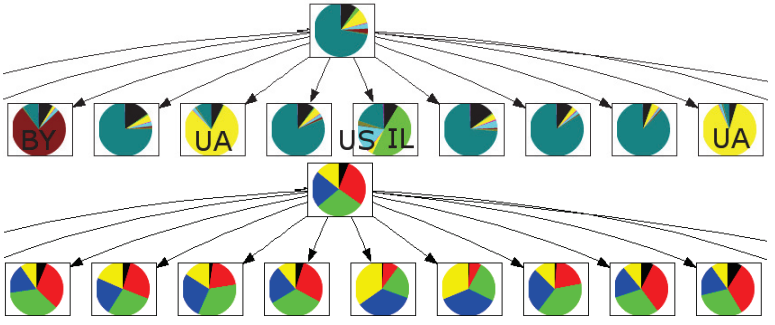


Figure 4.11: Subpartition of the Russian cluster, with the distribution of location (top) and age (bottom). Primarily Russian clusters (#2, 4, 6 and 8) have characteristic age distribution difference.

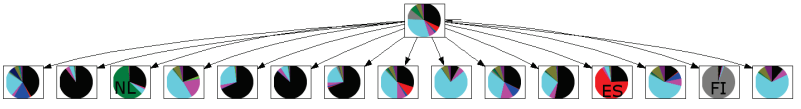


Figure 4.12: Location distribution of the subpartition of the International/Europe cluster. Black pie denotes locations other than the top 10. Since the parent cluster contains near 1/3 US location, several subclusters (#4, 8, 9, 13 and 15) still have a majority of US users (light pie). Certain countries have characteristic clusters; we marked the largest ones NL, ES and FI.

also Table 3.2), these include English-speaking clusters US, CA, UK, AU; the Russian cluster RU, BY, UA, EE, LT, IL; as well as non-English and Russian speaking characteristic countries BR, DE, ES, FI, NL, PH, SG.

In the top-level partitioning (Fig. 4.9) we see the characteristic Russian cluster [81, 165] as well as two international clusters, one with European connection, the other with mostly English speaking countries. The European cluster splits further by location with NL, ES, FI and several other characteristic countries (Fig. 4.12). The English-speaking cluster (Fig. 4.10) consists of UK, CA and AU; in addition they are clustered together with SG and PH. Finally the Russian cluster (Fig. 4.11) splits again by location with a US-IL partition also dominantly appearing.

Location within US is also known to be important. Unfortunately state information is completely missing from our recent crawl. We tried to reconstruct location within US by considering interest in California, San Francisco, New York, Chicago, Los Angeles or Boston. While we see indications of effect within the US sub-clusters, results are less indicative due to the lack of a more-or-less complete state information. We also observe strong international bounds across English-speaking countries except for the US-only clusters predominantly consisting of young people.

Age

Age is closely related to interest as noted in [101]. We bin ages after discarding people younger than 16 and older than 30 as in Fig. 4.9, middle. We see three clusters (No. 2, 3 and 5) with users predominantly from US; these clusters are distinguished by the age distribution. When considering splits in the lower level of the hierarchy, we also see clustering by age within a given location as soon as the location becomes dominant within a cluster. For example SG (Fig. 4.10) and RU (Fig. 4.11) users are already partitioned into two by age on the second level.

Polarized opinions

Polarization over political views [4] is examined by several authors. In our experiment we manually selected interest from user profiles that may be related to polarized opinion as in Table 4.2. While US parties did not show effect on principal axes, religion and atheism is characteristic in the LiveJournal Friends network. We selected users expressing Jesus, Jesus Christ, Atheism and Paganism as interest.

The top-level partitioning (Fig. 4.9) defines three US clusters, two with predominant interest in Jesus, while the third with Jesus in minority compared to Paganism and Atheism (this last cluster is also more international). In Fig. 4.9 we may also notice the apparent correlation with younger age and interest in Jesus. We note that certain clusters such as the Russian one are underrepresented for this type of interest.

Table 4.3: The running time, entropy, purity, normalized modularity and cluster ratio over the three real data sets. Cluster ratio is shown multiplied by 10^6. We test four algorithms: SVD with small component redistribution heuristic only, with all heuristics, semidefinite relaxation (SDP) and SDP with core and tentacle removal.

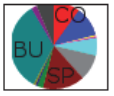
LiveJournal	runtime	entropy	purity	n.mod.	c.ratio
SVD redist only	1980m	0.105	0.812	2339	2
SVD all preproc	150m	0.073	0.853	2561	8
SDP no preproc	1755m	0.111	0.857	272	6
SDP all preproc	675m	0.072	0.854	2537	4
Telephone	runtime	entropy	purity	n.mod.	c.ratio
SVD redist only	80m	0.263	0.653	257	15
SVD all preproc	87m	0.239	0.648	206	12
SDP no preproc	2520m	0.237	0.634	237	14
SDP all preproc	2865m	0.252	0.628	251	13
UK-WEBSPAM	runtime	entropy	purity	n.mod.	c.ratio
SVD redist only	3m	0.362	0.199	35.69	116
SVD all preproc	5m	0.277	0.416	101.14	238
SDP no preproc	45m	0.266	0.426	51.77	864
SDP all preproc	47m	0.277	0.410	82.38	208

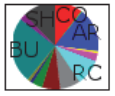
4.3.3 The UK2007-WEBSPAM host graph

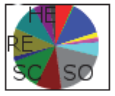
Over this host graph the k-way partitioning algorithm with $k = 15$ and $d = 30$ produced 100 clusters with three giant clusters remaining unsplit as seen in Fig. 4.13. In contrast, the Divide-and-Merge algorithm was not able to construct a single split, even with `ratio-init` $= 8$. The distribution of the 14 categories are shown in the pie charts. The first cluster has a very low fraction of known labels, most of which belongs to business (BU), computers (CO) and sports (SP), likely a highly spammed cluster. The second cluster has high ODP reference rate in business (BU), shopping (SH), computers (CO), arts (AR) and recreation (RC). Finally the largest cluster an opposite topical orinetation with high fraction of health (HE), reference (RE), science (SC) and society (SO). Among the less frequent four more categories, this latter cluster has a high fraction of kids and home while the second cluster contains games; news is negligible in the three clusters.

We experimented with various edge weighting schemes for the host graph, all of which resulting in roughly the same largest clusters that could not be further split as in Fig. 4.13, left. With an initial purity and entropy of 0.18 and 3.4, 100 clusters using the logarithm of edge weights resulted in purity 0.29, entropy 0.45, cluster ratio 0.00024 and normalized modularity -20.9, improved to 0.31, 0.44, 0.00026 and -17.9 by Jaccard weighting.

hosts	labeled
23595	2622
13111	8754
38279	14964
111149	35814







AR	arts
BU	business
CO	computers
HE	health
RC	recreation
RE	reference
SC	science
SH	shopping
SO	society
SP	sports

Figure 4.13: The size of the three largest remaining clusters and the number of labeled hosts within the cluster and in the entire crawl (bottom) as well as the distribution of categories within these clusters in the same order, left to right, with the list of abbreviations (left).

4.4 Conclusion and bibliographic notes

The results in this Chapter appeared in [104, 106] where I devised most of the heuristics. The first paper is cited by [61, 120, 167].

We gave a k-way hierarchical spectral clustering algorithm with heuristics to balance cluster sizes. We also implemented the heuristics in the recent Divide-and-Merge algorithm [43]. Our algorithm outperformed Divide-and-Merge for clustering the telephone call graph. We also measured the effect of several choices for the input to SVD: we found the weighted Laplacian performing much better than the unweighted counterpart and introduced a neighborhood Jaccard weighting scheme that performs very good for SVD input.

We demonstrated that spectral graph partitioning can be performed on very large power law networks after appropriate preprocessing heuristics. Our preprocessing steps include the removal of densely connected communities that are of small size on the global scale as well as the contraction of long "tentacles", loosely connected users that form large chains out of the center of the network.

Our central findings are related to the comparison of the SVD vs. semidefinite programming relaxation of the graph partitioning problem [109]. We show the SVD based partitioning quality can be improved to at least as good as the semidefinite one with large gains in speed. In particular the Lanczos algorithm based SVD can be parallelized since it consists of the multiplication of a vector with the input Laplacian. In addition SVD has good approximate solutions [59, 142]. In future work we plan to test distributed approximate SVD for very large graphs such as the UK2007-WEBSPAM page level graph with over 3 billion edges.

Of independent interest is our top-level analysis of the LiveJournal blogger Friends network, a data set of over three million users, in near 80% from US, 6% from Western Europe and 5% from Russia and East Europe. Here the components reveal global aspects of the network such as location, age, or religious belief. In future work more types of interest can be analyzed and the techniques presented here can be applied to blog posts or other large social networks.

Chapter 5

Generative Models of Hard-to-Partition Social Networks

In this Chapter we investigate the hardness of clustering synthetic graphs constructed by various social network models. Spectral graph partitioning has recently gathered bad reputation for failure over large scale social networks. While Lang [109] in part suggests this may be due to the expansion and the power law degree distribution in these networks, in our experiments graphs generated by known models for social-like networks, the preferential attachment model of Barabási et al. [18], the evolving copy model of Kumar et al. [100] and the small world model of Kleinberg [97] are all easily partitioned in a balanced way by the spectral method.

In our experiments the existing models are insufficient to explain the failure of spectral partitioning. As a main reason we find dense communities interconnected by long tentacles. We call a subgraph *tentacle* if it can be built by recursively adding low degree nodes. A tree is an obvious example of a tentacle; we may however have cycles or even somewhat wider objects built by degree 3 or higher nodes in a tentacle. Notice that our notion of a tentacle is reminiscent to the octopus structure described by Lang [109], although key is that the tentacles connect a large number of dense regions. The dense regions are seemingly similar to the dense bipartite communities described by the evolving copy model [100]; surprisingly however this model does not generate sufficiently dense communities needed for the observed bad behavior of spectral partitioning.

We construct a new combined model that generates graphs hard to partition with the spectral method and shows the observed size distribution of dense communities and tentacles. We first model densely settled regions in the Kleinberg geographic small world model [97] over a 2D grid by selecting nodes uniformly at random and replacing them by small cliques. Then we prescribe a power law degree distribution over this graph and generate the given number of edges independent with probability proportional to the Euclidean distance in the underlying grid. Certain properties such as the number of dense regions and the distribution of tentacle

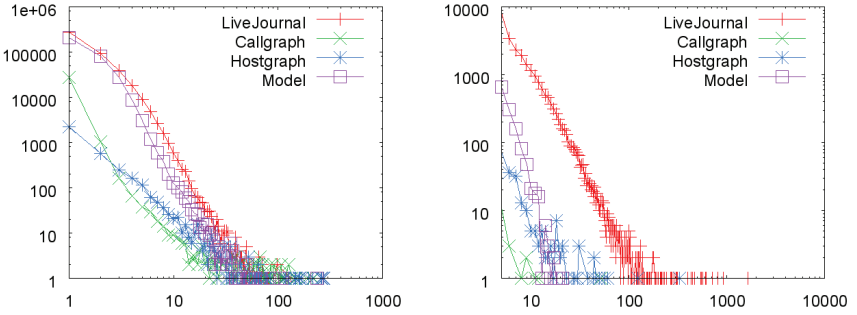


Figure 5.1: Left: the distribution of the size of the tentacles identified by Algorithm 4.2.5. Right: the distribution of the size of the communities identified by the modified SCAN Algorithm 4.1.2. Both charts are on the log-log scale and the horizontal axis shows the size of the component while the vertical the number of components with that size.

sizes observed in graphs generated by this model are very similar to the hard instances for spectral partitioning.

Network models such as the preferential attachment [18], evolving copy [100] or Kleinberg's small world [97] describe certain properties of social networks and Web graphs such as the degree distribution, low diameter, geographic concentration of the contacts and even certain dense communities.

We show that the above models however do not explain the hardness of clustering. In what follows we describe our procedures to generate graphs according to these models and also give our new model based on Kleinberg's small world [97] combined with power law degree and community distributions. In Fig. 5.1 we show the distribution of the sizes of community cores and tentacles in the models as well as the real graphs in Section 3.1. And in Table 5.1 we show the 15th largest singular value under different heuristics as an indicator of the hardness for partitioning.

5.1 The Barabási-Albert model

While the simplest random graph model, the Erdős-Rényi (ER) model [30, 63] assumes that edges are chosen independent at random for a graph with a fixed vertex set. Most real world networks are however open and new nodes may get in and out of it. The ER model assumes that two nodes are connected by uniform distribution. Experiment on real graphs show that the edges have a "preferential attachment" property, new nodes are more likely to connect to popular, high-degree nodes.

Barabási and Albert [16, 17, 18] propose a growing model, where the degree-distributions correspond to observations. The BA model is defined as follows:

Growing Start from a few (m_0) node, then in every timeperiod add some new node with $m \leq m_0$ edge, that connect to old nodes. The nodes are labeled by age, the degree of the i-th is k_i.

Preferential attachment The probability to connect to the node i is

$$C \cdot k_i \Big/ \sum_j k_j$$

where C is the growth constant.

After t time this will lead to a network, that contains $N = t + m_0$ node, and $m \cdot t$ edge. The degrees will follow a $P(k) \sim k^{-\gamma}$ like distribution, where $\gamma_{BA} = 3$. The scale exponent is independent from m, the only parameter. Also P_k is independent from time, or the size of network, so the model is scale-free.

We can observe that the older nodes gather more degree than new ones. A node that becomes popular will be even more popular.

This model generates power law degree distribution with the exponent -3; if random noise is added to the edge selection procedure, we may obtain different exponents as well [136]. By generating graphs according to these models we obtain neither cores nor tentacles and all such graphs can be partitioned by the basic spectral method.

5.2 The Evolving Copy Model

In the evolving copy model of Kumar et al. [100] whenever new vertices arrive, they select an old vertex uniform at random and copy their edges with noise.

The parameters of the model are $\alpha \in (0, 1)$ copy-factor and $d \leq 1$ outdegree factor. Let u be a new node, its outdegree d_u. The outedges of u are generated in the following way:

- Choose an existing p node uniformly random as prototype.
- For each $i \in 1..d_p$ for the i-th edge with probability α connect to another node chosen uniformly random, with probability $(1 - \alpha)$ copy the i-th edge of p.

Two variants of this model exist, the first one adds one new node to the network in every time period. The second one adds k new nodes to the network where k is proportional to the network size. While we observed no significant difference between the two variants with respect to clustering hardness, all experiments in this section belong to the latter one.

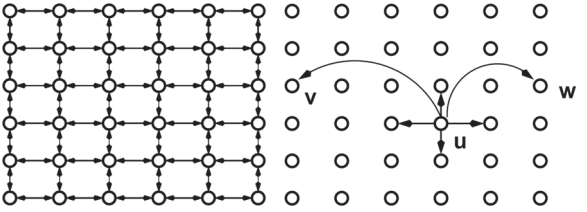


Figure 5.2: Left: the local contacts. Right: the distant contacts.

In addition to achieving power law degree distribution, the graphs in this model have a large number of dense bipartite cliques, a property characteristic to Web graphs and a possible cause of dense regions that could result in imbalanced spectral clusters. This model already generates hard instances for spectral partitioning, however they can be resolved by tentacle removal itself (Table 5.1). We observe no cores; tentacle size distributions are similar to the hard instances (Fig. 5.1).

5.3 Kleinberg's Small World Model

The small world graph model of Kleinberg [97] captures a different property of social networks, namely the fact that short paths don't just exist but they can be found efficiently by using only local information. In his model there is an underlying 2D grid. Kleinberg defines *lattice distance* between two nodes *(i,j)* and *(k,l)* to be the number of "lattice steps" separating them: $d((i,j),(k,l)) = |k-i|+|l-j|$. For the constant $p \geq 1$, the node u is connected to every other node within lattice distance p, these are its local contacts. Nodes also select a constant $q \geq 0$ number of distant neighbors in the grid. The i^{th} edge of u is connected to v with probability proportional to $[d(u,v)]^{-r}$ where r is another input parameter of the model.

It has been shown that with the choice of $r = 2$ routing is possible in the network quickly. For smaller values than 2 the connections become to local, for greater values the network becomes to random.

For an appropriate good choice of q, a medium number of edges are generated from each node, then, similar to the evolving copy model, tentacles appear and partitioning is possible only after contracting them (Table 5.1). This model generates no cores.

Table 5.1: The 15th largest singular value for different input and the choice of the heuristics for tentacle contraction (tent) and core removal (SCAN). Figures in boldface denote cases when no balanced partitioning is possible at the first split by Algorithm 4.2.1.

σ_{15}	plain	tent.	SCAN	both
Kumar	**0.956**	0.783	**0.956**	0.783
Kleinberg	**0.980**	0.811	**0.980**	0.811
New Model	**0.997**	**0.994**	**0.988**	0.810
LiveJournal	**0.999**	**0.989**	**0.993**	0.987
Telephone	0.897	0.886	0.897	0.881
UK2007-WEBSPAM	**0.894**	0.856	0.867	0.698

5.4 A model for hard-to-cluster networks

Our new model is a power-law-degree-and-clique small world, defined as follows. The starting point is the small world graph model of Kleinberg [97] with nodes placed over a 2D grid[1]. Next we generate geographically dense regions over the grid by assigning density to each node according to a power law distribution with exponent -3. Finally as in Kleinberg's model we connect nodes with probability inversely proportional to their squared Euclidean distance. However in Kleinberg's model the degree is constant; in our model for each vertex we generate a number t by a power law distribution with exponent -3 and add t edges independent with probability as in Kleinberg's model.

In this new model, as seen in Table 5.1, spectral partitioning produces balanced enough partitions to pass the small component redistribution heuristics (Algorithm 4.2.1) only after both dense community removal and tentacle contraction. The distribution of community and tentacle sizes follow close power law very similar to those of the real graphs and in particular to LiveJournal, the hardest instance.

We remark that a simpler version itself suffices as a hard example for spectral partitioning. Instead of a power law density generation, we may simply select roughly 1% of the grid points and add 10 element clusters to these points. The tentacle size distribution remains the same and spectral partitioning remains hard. We also remark that power law and log-normal distributions are similar in their heavy tail; a power law community size distribution may hence follow from the log-normal settlement size distribution as seen in Fig. 4.3.

5.5 Conclusion and bibliographic notes

The results in this Chapter appeared in [102] in a joint work with my Thesis advisor.

We showed that graphs generated by existing social network models are not as difficult to cluster as they are in the real world. For this end we gave a new combined model that yields

[1]To simplify generation we in fact used a 2D torus.

degenerate adjacency matrices and hard-to-partition graphs.

Chapter 6

Large Scale SVD with Missing Values in Recommenders

Recommender systems predict the preference of a user on a given item based on known ratings. In order to evaluate methods, in October 2006 Netflix provided movie ratings from anonymous customers on nearly 18 thousand movie titles [25].

In this chapter we concentrate on recommenders based solely on low rank approximation and compare various implementations and parameter settings. The low rank approximation of the rating matrix as a recommendation is probably first described in [28, 84, 138, 144] and many others near year 2000.

The key difficulty in computing the low rank approximation lies in the abundance of missing values in the rating matrix: the Netflix matrix for example consists in 99% of missing values. While several authors describe expectation maximization (EM) based SVD algorithms dating back to the seventies [75] and [39, 150, 169] describe the method for a recommender application, we are aware of no systematic studies on large scale problems. In particular all these results consider small, few thousands by few thousands submatrices of the EachMovie or Jester databases, of several orders of magnitude smaller than handled by our algorithms.

A successful approach to a low rank recommender is described by Simon Funk in [74] is based on an approach reminiscent of gradient boosting [72]. The algorithm opens a number of theoretic questions including its relation to published results that solve SVD with missing values as well as the effect of the parameters on convergence speed and overfitting. One of the main intents of this Chapter is to understand the relation of his method to existing missing value SVD approaches.

We note that several more advanced results appeared after our result, including the matrix factorization methods of Paterek [134] and Teams Gravity [152] and BellKor [99] of the Netflix Prize competition.

6.1 Introduction

In this chapter, our main contributions are:

- The implementation of SVD based recommenders for large scale problems with specific attention to the scalability issues of handling full matrix imputation values. Note that previous results except for [74] handle data of several orders of magnitude smaller than ours.
- The application of these methods for the KDD Cup 2007 Task 1, the prediction whether a user rated a given movie in a given period of time.
- The comparison of various methods in terms of recommendation accuracy and convergence rate, with emphasis on the explanation of parameters that speed up convergence.

The rest of this Chapter is organized as follows. In the rest of the Introduction we describe related approaches, the experimental setup, and the SVD algorithm implementation used. In Section 6.3 we measure the effect of filling missing values by zeroes, by averages and finally by the output of an item-item similarity based recommender. This is a challenging task since the full recommendation matrix has several billion entries.

After imputation by external values we turn to EM approaches that compute a low rank approximation in one iteration and impute the outcome for a next iteration. In Sections 6.4 and 6.5 we use the Lanczos algorithm and power iteration, respectively. In both cases we resolve the implementational challenge of handling the full matrix arising by the previous iteration. We observe slow convergence of the methods; we evaluate methods to speed up by combining partial results.

Finally in Section 6.6 we describe a least squares based EM approach to directly optimize a low rank solution for small error. In this algorithm we optimize each user vector separately, thus enabling a user-by-user adaptive control on the number of dimensions used. Our key observation is that for a user with r ratings, roughly $r/25$ dimensions should be used.

In all cases we investigate the effect of the dimensionality of the low rank approximation; we observe best performance at a few dimensions and overtraining (good performance on the training but deterioration on the test (probe) set) as the number of dimensions approaches 100. All methods are compared in Section 6.7.

6.1.1 Data set, evaluation and experimental setup

Netflix provided over 100 million ratings from n over 480 thousand randomly-chosen, anonymous customers on m nearly 18 thousand movie titles [25]. The company withheld certain portion of the ratings as a competition qualifying set that we will not use in this report. Netflix

also identified a *probe* subset of the complete training set; we refer the remaining known ratings as the *training* data.

We use the *root mean squared error*

$$\text{RMSE}^2 = \sum_{ij \in R} (w_{ij} - \hat{w}_{ij})^2 \tag{6.1.1}$$

as the single evaluation measure, where w_{ij} is the actual rating, an integer in the range 1–5, given by user i to movie j, and $\hat{w}_{ij}$ is the prediction given by the recommender system. We present RMSE values for the train and the separated test set but not the qualifying set.

The experiments were carried out on a cluster of 64-bit 3GHz P-D processors with 4GB RAM each and a multiprocessor 1.8GHz Opteron system with 20GB RAM.

6.1.2 Related work

Recommenders based on the rank k approximation of the rating matrix based on the first k singular vectors are probably first described in [28, 84, 138, 144] and many others near year 2000.

The Singular Value Decomposition (SVD) of a rank ρ matrix W is given by $W = U\Sigma V^T$ with U an $m \times \rho$, Σ a $\rho \times \rho$ and V an $n \times \rho$ matrix such that U and V are orthogonal. By the Eckart-Young theorem [80] the best rank-k approximation of W with respect to the Frobenius norm is

$$||W - U_k \Sigma_k V_k^T||_F^2 = \sum_{ij} (w_{ij} - \sum_k \sigma_k u_{ki} v_{kj})^2. \tag{6.1.2}$$

where U_k is an $m \times k$ and V_k is an $n \times k$ matrix containing the first k columns of U and V and the diagonal Σ_k containing first k entries of Σ.

The RMSE differs from the above equation only in that summation is over known ratings

$$\text{RMSE}^2 = \sum_{ij \in R} \text{err}_{ij}^2 \text{ where } \text{err}_{ij} = w_{ij} - \sum_k \sigma_k u_{ki} v_{kj} \tag{6.1.3}$$

where R denotes either the training or the test set. To simplify notation we extend err_{ij} with value 0 for $ij \notin R$.

As already emphasized in one of the early works [144], the crux in using SVD for recommenders lies in handling missing values in the rating matrix W. Goldberg et al. [78] for example require a gauge set where all ratings are known, an assumption clearly infeasible on the Netflix data scale. Azar et al. [13] prove asymptotic results on replacing missing values by zeroes and scaling known ratings inversely proportional to the probability of being observed.

The *expectation maximization* algorithm proceeds as follows. Given the output U_k, Σ_k and V_k matrices of sizes $m \times k$, $k \times k$ and $n \times k$, respectively, produced by SVD in the maximiza-

tion step, the expectation step produces a matrix with entries

$$\begin{aligned} \hat{w}_{ij} &= \begin{cases} w_{ij} & \text{if } ij \in R \\ [U_k \Sigma_k V_k]_{ij} & \text{otherwise} \end{cases} \\ &= \text{err}_{ij} + [U_k \Sigma_k V_k]_{ij} \end{aligned} \tag{6.1.4}$$

where the last equality follows by the definition of err as in (6.1.3). The algorithm alternates between SVD computation (maximization) and the expectation equation (6.1.4) until convergence. It is easy to see that U, Σ and V minimizing (6.1.3) is a fixed point of this iteration; up to our best knowledge, this is the only theoretical result known about the convergence properties of the above EM missing value SVD algorithm. While in our implementation k is typically fixed as input parameter, variants of this algorithm may increase or even decrease k as the iterations proceed.

This algorithm is perhaps first used for recommenders by Canny [39] and then several others [150, 169]. Canny [39] concentrates on privacy issues; he reports experiments on much smaller scale such as a subset of the EachMovie data. Srebro and Jaakkola [150] compare methods that fill missing entries by zeroes, also by scaling known entries as in [13], using a gauge set as in [78] as well as a variant of the EM procedure. They also give a number of hints related to the convergence of the EM method. First of all they observe the algorithm may reach local optimum; it is unclear whether this may happen in the missing value case as well. They also show that different rank solutions are non-orthogonal; for this end they propose starting out with a large rank approximation and gradually reduce the rank in the EM iterations.

The EM algorithm for solving SVD with missing values dates back to the seventies; [76] gives a more recent description. In early results, the generic idea of filling missing values by expectation maximization to our knowledge appears first in [87] and is perhaps best described by [51] with the explicit mention of factor analysis as an application but apparently no references between another line of work [75, 141]. To our knowledge, the first paper that presents the missing value problem is [141]; [75] generalizes the missing value problem to a weighted regression and solves it by EM.

More recently several authors reinvented EM for SVD. In [33] the idea of representing the missing data imputation matrices by their known SVD U and V appears that is key in our sparse implementation. Zhang et al. [169] give an approximate SVD algorithm with theoretical analysis, however tests are only shown on small scale data.

Theoretical works on SVD based recommenders exist [57] but we are aware of none that address the missing value problem. In particular we aware of no results on the convergence except for the negative experimental findings of Srebro and Jaakkola [150].

Dimensionality reduction is investigated for gene expression data as well. Several authors

Algorithm	$k = 10$	$k = 15$	$k = 20$	$k = 25$	$k = 30$	$k = 50$
Lanczos	1:58				7:40	
Power	1:57				5:50	
Adaptive	33.7	33	31	29	28	25

Table 6.1: Running times in the form of minutes or hours:minutes for a single iteration over the Netflix Prize matrix with n over 480 thousand, m nearly 18 thousand and over 100 million nonzeroes. For Lanczos and Power the top column k gives dimensionality while for Adaptive the dimensionality is $1 + r/k$ for a user with r ratings, i.e. here unlike for the other two algorithms the running time decreases with k. For Lanczos we use 40 iterations altogether. For Power we use 100 for a single dimension, hence here we get linear dependency on k.

[88, 154] compare imputation methods including nearest neighbors as well as the EM approach with controversial findings for accuracy but a definite identification of the very slow convergence for EM.

6.1.3 SVD implementation

In our implementation we used the Lanczos code of `svdpack` [26] and compared it with a power iteration developed from scratch (See Section 4.3.1). Running times are shown in Table 6.1.

We also measure the number of dimensions of the approximation. Typically, in SVD the use of dimensionality is restricted by efficiency considerations, and for example for spectral clustering [10, 113] suggest that more eigenvalues produce better quality cuts. However we observe that as the number of dimensions increase beyond roughly 10, we overtrain and prediction quality deteriorates; for this reason we also test an algorithm that adaptively selects more dimensions for users with more ratings in Section 6.6.

6.2 KDD Cup 2007

6.2.1 Problem description

We present our first place winner method for Task 1 "Who Rated What in 2006". The task was to predict the probability that a user rated a movie in 2006 (with the actual date and rating being irrelevant) for a given list of 100,000 user–movie pairs of the Netflix Prize data set [25] . The users and movies are drawn from the Prize data set, i.e. the movies were released (or at least received ratings) before 2006 and the users also gave their first rating before 2006. In addition, none of the pairs selected for the KDD Cup task were rated in the training set.

Our method is summarized as follows:

1. The combination of separate estimates for the number of ratings for each movie and

each user by a naive user–movie independence assumption. Our movie ratings prediction uses time series analysis aligned with movie and DVD release dates from the IMDB and `videoeta.com` databases. User rating numbers are on the other hand reconstructed from sample margins.

2. The implementation of an SVD (Section 6.2.2) and an item-item similarity based recommender as well as association rule mining.

3. Method fusion by using the machine learning toolkit Weka [160].

We use the *root mean squared error* equation (6.1.1) as the single evaluation measure, where w_{ij} is a 0–1 matrix with value 1 if user i gave rating for movie j, and $\hat{w}_{ij}$ is the predicted value in the range of 0 to 1 given by the recommender system.

The use of RMSE implies that we actually predict the probability of the existence of a rating: for a random variable that has value 1 with probability p and 0 otherwise, the RMSE of the prediction of its value is minimized by p. If we correctly guess the number of ratings 7,804 in the 100,000 sample, then this method results in an RMSE of 0.268 that would reach 5-6th place in the Cup, indicating the hardness of correctly predicting this value. Notice however that the RMSE of 0.279 of the trivial all zeroes prediction would also reach 10-13th place and remains not very far from the winner RMSE.

With certain variation depending on parameter settings, the running time range was 15 minutes for SVD, few hours for the item-item similarity based recommender, few days for frequent patterns and finally few minutes for linear regression. Mining frequent patterns turned out to be the most time consuming. We present a more thorough experimentation with frequent patterns that we could not afford for the KDD Cup competition due to CPU time limitations.

6.2.2 SVD based recommendation

For training we use the full 0–1 matrix of all known ratings; the rank k approximation of the matrix yields our prediction.

While the Frobenius norm is simply the RMSE of the prediction for the existence of the rating if the user–movie pairs are selected uniformly at random, this is not true for the sampling method used for producing the Task 1 pairs. If the probability that the pair formed by user i and movie j is selected in the sample is p_{ij}, then we have to minimize

$$\begin{aligned} \sum_{ij} p_{ij} \cdot (w_{ij} - \sum_k \sigma_k u_{ki} v_{kj})^2 \quad = \\ \sum_{ij} (\sqrt{p_{ij}} \cdot w_{ij} - \sqrt{p_{ij}} \cdot \sum_k \sigma_k u_{ki} v_{kj})^2, \end{aligned} \tag{6.2.1}$$

which is minimized similarly by the SVD of $\sqrt{p_{ij}} \cdot w_{ij}$, divided pointwise by $\sqrt{p_{ij}}$.

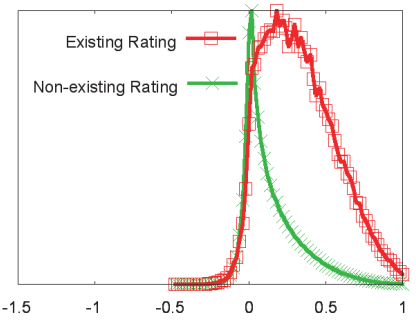


Figure 6.1: The distribution of the 10-dimensional approximation of the user–movie matrix for pairs with, respectively without ratings.

In our implementation we used the Lanczos code of `svdpack` [26] that turned out both the fastest and the most precise in our recent experiments [103, 104]. Since we observed overfitting for larger number of dimensions [103] we used the 10-dimensional approximation of the scaled matrix as in equation (6.2.1) where the p_{ij} values are those obtained by naive user–movie independence assumption. The difference between the distribution of the predicted value for the actual ratings, respectively no-ratings is seen in Fig. 6.1.

6.2.3 Results

We combined the four predictions of the naive independence, SVD, item-item correlation and association rule based approaches by the linear regression method of the machine learning toolkit Weka [160]. We obtained the equation

$$\begin{aligned} 0.5533 \cdot p_{um} \quad &+ \\ 0.029 \cdot \text{correlation} \quad &+ \\ 0.1987 \cdot \text{SVD} \quad &+ \\ -0.0121 \cdot \text{assoc\_rules} \quad &- \quad 0.0042 \end{aligned}$$

as the final prediction that reaches RMSE 0.256, gaining 0.007 over the first runner up and 0.023 over a pure all zeroes prediction.

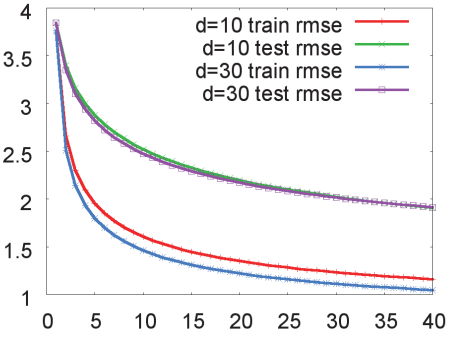


Figure 6.2: The RMSE as the function of the iterations for the simple Lanczos EM algorithm, the missing values are filled with zeros in the initial matrix.

6.3 Missing data imputation from external results

In the simplest approach we use external sources of data to fill missing ratings and optimize for error in Frobenius norm as in equation (6.1.2) in the hope that external data fit well and optimization for Frobenius yields good approximation for the RMSE equation (6.1.3) as well. First, as expected, we show filling missing data with zeroes as suggested for example by [13] badly fail over the rare Netflix data by providing recommendations near 0 due to the abundance of zeroes in the matrix after imputation. We improve performance first by using averages, then by the outcome of a more sophisticated recommender based on item-item similarities. Surprisingly user averages perform better than the output of the recommender in this case.

Imputation by zeroes and averages are fairly straightforward given control over data access within the SVD algorithm as described in Section 6.1.3. It is however a challenging question for a full recommendation matrix that we describe in Section 6.3.1.

We show RMSE values for imputation with zeroes and averages as the first iterations in Fig. 6.4. We observe very poor performance; in particular by filling with zeroes we are so far off from optimum that even a large number of EM iterations remain insufficient to converge.

6.3.1 Output of an item-item similarity based recommender

We implemented the adjusted cosine similarity [145] for an item-item similarity based recommender that recommends an unrated movie j to a given user i by the weighted average of the nearest N movies to i rated by the user. Here N is a parameter; roughly speaking, this approach increases the fraction of known values by a factor of N.

The Lanczos implementation of `svdpack` [26] accesses the matrix by in one step computing a product of a vector with either the matrix or its transpose. The SVD implementation may hence access the nearest neighbor lookup table whenever a matrix multiplication is needed. The implementation requires space to store the rating matrix and the nearest neighbor index. In the running time however the matrix multiplication time becomes dependent on the size of the full matrix mn instead of the much smaller number of known ratings. While this implementation is memory efficient, it is so slow that we had to give up tests in this direction.

We may however give an efficient item-item similarity based imputation by slightly regressing the item-item similarity based output towards the user average $\hat{u}_i$, as follows. We form the submatrix S of the item-item similarities where for efficiency considerations we only keep the top 100 largest entries in each row. We even discard those of the values below 0.5. When predicting a rating for user i and movie j, we then compute the sum of $w_{ij'} - \hat{u}_i$ weighted by the similarity of j and j' for all j' where both the similarity and the rating $w_{ij'}$ are known. Next in order to give a prediction we have to add the normalized value to $\hat{u}_i$. In order to be efficiently computable, we simply normalize by 100, even though typically there are less than 100 j' terms in the sum and their similarity values may be as low as 0.5.

The algorithm proceeds as follows. We let H be an $n \times m$ matrix where each row contains the user averages $\hat{u}_i$ as identical values and

$$F_{ij} = \begin{cases} (w_{ij} - \hat{u}_i)/100 & \text{if } (ij) \in R \\ 0 & \text{otherwise.} \end{cases}$$

The product of vector x with the imputed matrix W' can be efficiently computed as

$$x \cdot W' = x \cdot H + x \cdot F \cdot S + x \cdot E$$

where

$$E_{ij} = \begin{cases} W' - H - F \cdot S & \text{if } (ij) \in R \\ 0 & \text{otherwise} \end{cases}$$

removes the effect of the similarity based recommendation where the actual rating is known. In Fig. 6.3 we see the RMSE for a 10-dimensional SVD started by these values.

6.4 Sparse Lanczos implementation within an EM framework

When using the Lanczos algorithm after an expectation step, we face the same difficulty of imputing a full matrix as in Section 6.3.1. We provide a similar solution below, with careful analysis of the number of operations used. Note that unlike in the previous section, we need

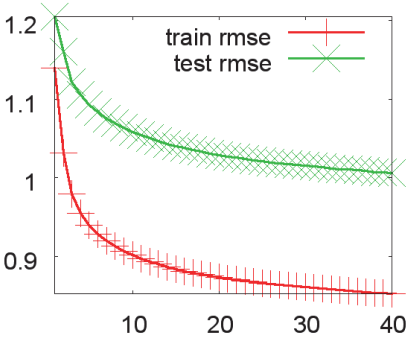


Figure 6.3: The RMSE as the function of the number of iterations for the simple Lanczos EM algorithm with the first iteration imputed with the output of the simplified item-item similarity based recommender.

a large number of iterations until convergence hence not just the space but also the speed of handling the dense input is crucial.

Since in a Lanczos iteration we require the product of vector $\mathbf{x} = (x_1, x_2, \ldots)$ with $\hat{w}$ (or similarly with its transpose), by the EM algorithm equation (6.1.4) we compute

$$\text{err} \cdot \mathbf{x} + U_k \Sigma_k V_k \cdot \mathbf{x}.$$

The space required by this algorithm is equal to $O(kn+km)$ for multiplying $\mathbf{x}$ with the imputed low rank approximation term by term from right to left, in addition to the number of non-missing values in the rating matrix. Hence the additional work due to imputation is negligible in the Lanczos computation.

6.4.1 Speeding up convergence

In order to speed up convergence we apply the generic method of finding an optimal linear combination of the values $\hat{w} = \sum_k \sigma_k u_{ki} v_{kj}$ in the current and $w^{(t-1)}$ in the previous iterations: we minimize the quadratic expression of λ in the RMSE equation (6.1.3):

$$\sum_{ij \in R} (w_{ij} - \lambda \hat{w}_{ij} - (1-\lambda) w_{ij}^{(t-1)})^2.$$

We then let $w^{(t)} = \lambda \hat{w} - (1-\lambda) w^{(t-1)}$ for the λ value at the minimum.

In the above naive form however matrix values in the next iteration will arise as a linear combination of a rank k matrix and the previous $w^{(t-1)}$, which in turn depends on $w^{(t-2)}$ and

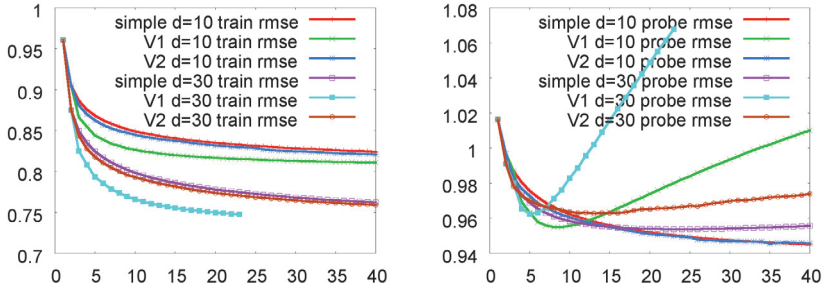


Figure 6.4: The RMSE as the function of the iterations for the simple Lanczos EM algorithm and the two convergence boosting variants. The results for the training set are on the left, and for the test set they are on the right. Curves correspond to different dimensionality with V1 and V2 denoting the two convergence boosting variants.

inductively on all previous partial results. Since it is infeasible to store either full matrices or all partial results, we have to relax the above algorithm. We give two versions next that obey the scalability requirements.

Our two implementations of linearly combining current and previous results use the last two low rank approximations $U_k^{(t)}$, $\Sigma_k^{(t)}$, $V_k^{(t)}$ and $U_k^{(t-1)}$, $\Sigma_k^{(t-1)}$, $V_k^{(t-1)}$. In the first algorithm we combine as

$$\lambda U_k^{(t)} \Sigma_k^{(t)} V_k^{(t)} + (1-\lambda) U_k^{(t-1)} \Sigma_k^{(t-1)} V_k^{(t-1)},$$

i.e. using only the low rank matrix instead of the combined iteration $t-1$ approximation. The minimum of the quadratic expression in λ is attained, with the notation of $W = U_k^{(t)} \Sigma_k^{(t)} V_k^{(t)}$ and $W' = U_k^{(t-1)} \Sigma_k^{(t-1)} V_k^{(t-1)}$, at

$$\lambda = -\frac{\sum_{ij \in R} (W_{ij} - W'_{ij})(W'_{ij} - w_{ij})}{\sum_{ij \in R} (W_{ij} - W'_{ij})^2}.$$

In the second variant we combine the low rank decomposition elements: from $U_k^{(t)}$, $V_k^{(t)}$ and the previous result we get $\hat{U}_k^{(t)}$ and $\hat{V}_k^{(t)}$. We ignore $\Sigma_k^{(t-1)}$ based on the observation that the Σ_k converges fast. With the simplified notation U_i and U'_i for the i-th row of $U_k^{(t)}$ and $U_k^{(t-1)}$, respectively and the same notation for the columns of the V, we minimize

$$\sum_{ij \in R} \left((\lambda (U_i - U'_i) + U'_i) \Sigma (\lambda' (V_i - V'_i) + V'_i) - w_{ij} \right)^2.$$

Here for each λ' there is a corresponding optimum λ given by

$$X_{ij} = \lambda'(U_i - U_i')(V_j - V_j') + (U_i - U_i')V_j',$$

$$\lambda = -\frac{\sum_{ij \in R} \lambda' U_i \Sigma (V_j - V_j') + U_i' V_j' - w_{ij}) X_{ij}}{\sum_{ij \in R} X_{ij}^2};$$

we select the best by substituting a low number of probe λ values.

6.5 Power iteration within an EM framework

For a full matrix W, power iteration proceeds by repeatedly letting

$$u^{(t+1)} = W^T \cdot v^{(t)} / ||v^{(t)}||, \tag{6.5.1}$$

$$v^{(t+1)} = W \cdot u^{(t)} / ||u^{(t)}||. \tag{6.5.2}$$

The algorithm converges to the first singular vectors also called the "hub" and "authority" vectors [98]; due to numeric errors this holds even if we start out with an initial $v^{(0)}$ orthogonal to the first vector $V_{\cdot 1}$ unless we *orthogonalize*, i.e. project each or some $v^{(t)}$ onto the hyperplane orthogonal to $V_{\cdot 1}$. By orthogonalization to the first $k-1$ singular vectors V_{k-1} however we may obtain the next $V_{\cdot k}$ by iteration (6.5.1–6.5.2).

In the presence of missing data we may use (6.5.1–6.5.2) in the expectation maximization framework by filling $ij \notin R$ by $w_{ij} = \sigma_1 u_i^{(t)} \cdot v_j^{(t)}$. First we observe that if $v^{(t)}$ is a good approximation of $V_{\cdot 1}$, then $||v^{(t+1)}|| \approx \sigma_1$, hence the iteration turns to

$$u_i^{(t+1)} = \sum_{j:ij \in R} w_{ji} v_j^{(t)} / \sigma_1 + \sum_{j:ij \notin R} {v_j^{(t)}}^2 \cdot u_i^{(t)}, \tag{6.5.3}$$

$$v^{(t+1)} = W \cdot u^{(t)} / ||u^{(t)}||. \tag{6.5.4}$$

This approach is split into two different heuristic implementations in the next two subsections. The RMSE for the basic implementation (6.5.3–6.5.4) is shown in Fig. 6.5.

6.5.1 Method of individual increments

We rewrite (6.5.3) as

$$u_i^{(t+1)} = \sum_{j:ij \in R} \left(\frac{w_{ji}}{\sigma_1} - v_j^{(t)} \cdot u_i^{(t)}\right) v_j^{(t)} + \sum_j {v_j^{(t)}}^2 \cdot u_i^{(t)}. \tag{6.5.5}$$

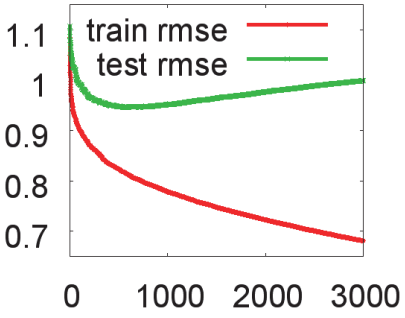


Figure 6.5: The RMSE as the function of the number of iterations for the basic power iteration method given by equations (6.5.3–6.5.4).

Given the assumption that the v are normalized that we may enforce in our algorithm, the second term is simply $u_i^{(t)}$. As a heuristic speedup, we split (6.5.5) into increments over u_i for individual j, replacing u_i by a new value in each step. This yields an algorithm with a cycle over

$$u_i \leftarrow u_i + (w_{ji}/\sigma_1 - v_j \cdot u_i)v_j \tag{6.5.6}$$

very closely reminiscent of Simon Funk's steps [74]

$$u_i \leftarrow (1 - \text{lRate})u_i + K(w_{ij} - \sigma_1 u_i v_j)v_j. \tag{6.5.7}$$

We use an idea similar to the convergence acceleration in Section 6.4: we multiply the increment in (6.5.6) by a factor δ that minimizes the RMSE of the approximation with modified u_i,

$$\delta = \sum_{j':ij' \in R} \left(w_{ij'} - \sigma v_{j'}(u_i + \delta v_j \text{err}_{ij})\right)^2.$$

The minimum is easily computed as

$$\delta = \frac{\sum_{j':ij' \in R} v_{j'} \text{err}_{ij'}}{\sum_{j':ij' \in R} v_{j'}^2} \sigma v_j \text{err}_{ij}.$$

Unfortunately this algorithm appeared to diverge due to numeric errors for ratings with very small err_{ij}. Best results are obtained by using a median value near $\sigma/100$ very close to that suggested by [74].

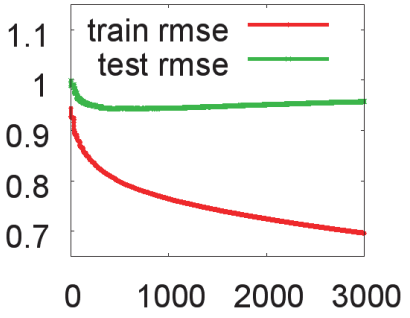


Figure 6.6: The RMSE as the function of the number of iterations for power iteration with individual increments given by equation (6.5.6).

6.5.2 Repeated hub and authority steps

If we repeat the "hub" iteration (6.5.1) more than once before an "authority" iteration (6.5.2), we observe no change in the full matrix case since the right hand side of (6.5.1) is independent of u. This is no longer the case for missing values however since imputation values depend on u. If we let

$$\Delta = \sum_{j:ij\in R} (w_{ij}/\sigma - u_i v_j)v_j \ , \quad v = \sum_{j:ij\in R} v_j^2$$

then t iterations of (6.5.3) give

$$u_i \leftarrow (1-h)^t u_i + (1-(1-v)^t)v^{-1}\Delta. \tag{6.5.8}$$

Best results are obtained in Fig. 6.7 if we use values $k = 5$ for the first singular vector computation and then decrease to 3, 2 and finally 1 for next dimensions. In addition we also combined this technique with individual increments as in (6.5.5):

$$u_i \leftarrow (1-v_j^2)^t u_i + (1-(1-v^2)^t)v^{-2}\Delta, \tag{6.5.9}$$

a formula again reminiscent of (6.5.7) of [74].

We may repeat the idea of the previous subsection and compute these steps individually for each i and j.

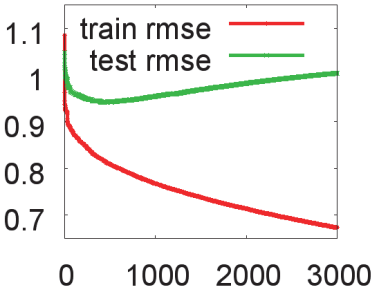


Figure 6.7: The RMSE as the function of the number of iterations for power iteration with repeated hub and authority steps given by equation (6.5.9).

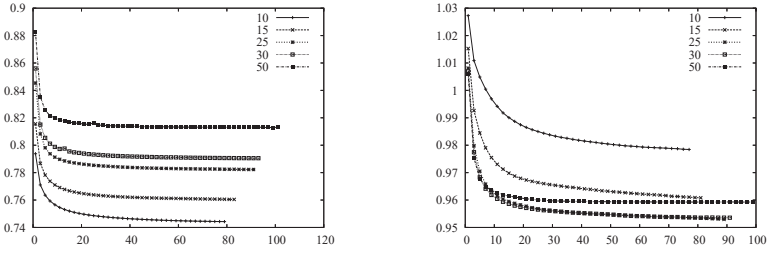


Figure 6.8: The RMSE as the function of the number of iterations for adaptive least squares given by equation (6.6.1). Different curves correspond to values of K where we compute least squares over the first $1 + r/K$ dimensions for a user with r ratings. **Left:** RMSE over the test set. **Right:** RMSE over the probe set.

6.6 A least squares approach with adaptive dimensionality

We give an algorithm that alternately computes and optimal U_k for a fixed V_k and then exchanges the role of U and V, similar to the "hub" (6.5.1) and "authority" (6.5.2) steps of the power iteration. For fixed V_k, optimizing the RMSE equation (6.1.3) can be done separately for the columns of U_k as

$$\sum_j (w_{ij \in R} - \sum_k \sigma_k u_{ki} v_{kj})^2 \tag{6.6.1}$$

as n regression problems.

The key idea in our algorithm is that once the regression is made separate for each user, we may adaptively select the right dimensionality for user i depending on the amount of ratings r given by her. Fig. 6.8 depicts RMSE for different values of constant K where we use the first $1 + r/K$ dimensions in the above expression. We observe overfitting on the training set: the more dimensions used, the better is the RMSE; however over the probe set values of K between 25 and 30 perform the best; for a large number of iterations apparently $K = 25$ takes lead.

6.7 Conclusion and bibliographic notes

The results in this Chapter appeared in [105] (KDD Cup) where I led the team and devised the SVD and combination methods as well as in [103] (missing value methods) where I devised most of the imputation methods. The first paper is cited by [94, 131, 132, 162] while the second by [71, 83, 107, 115, 139, 151, 153, 163, 164, 170]. As being one of the early results for matrix factorization, our methods were later outperformed by the methods of the Netflix Prize participants [99, 152].

In our findings the best method is Lanczos with 10 dimensions. Unfortunately the iterations are relative costly and convergence boosting approaches tend to give minor improvements only. Power iteration based methods, though performing very similar steps as Funk's algorithm [74], tend to overfit the training set. We believe more careful tuning could improve performance. The runner up is the adaptive dimensionality least squares approach.

The most carefully tuned implementation of Funk's algorithm (6.5.7) [74] reaches an RMSE slightly below 0.92 on the probe set with 95 dimensions in over 100 iterations and $K = 0.015$, lRate $= .001$. By setting $K = 0$ performance similar to ours is reported. We believe a thorough measurement over our algorithms might find improvements, however our main goal here was to understand the behavior of the missing value problem by investigating a large number of related algorithms.

For further work we propose the implementation and comparison of fast SVD approximations and experiments with graphs of even larger scale. We also plan to mix results, a method

that is known to yield significant improvement and in addition sometimes prefer weaker recommenders and thus slightly redraw the picture.

Chapter 7

Conclusions

We have surveyed some results of social network modeling and analysis as well as recommender systems with illustrations over the call logs of major Hungarian telephone companies with millions of users, the LiveJournal friends network, Web host graphs as well as the Netflix movie ratings data set.

We have considered real networks from the point of view of data mining, a new discipline that builds on results from machine learning, modeling and algorithmics with emphasis on data scale. Improved hardware capabilities enable the production of huge data volumes. In order to scale to the new demands, traditional problems require new algorithms and lead to new empirical findings.

Our key results also point out to the importance of data mining methodologies in network analysis. A generic data mining process starts with the appropriate choice for data preparation, cleansing and modeling. Given the results provided by the final algorithm, we may have to reiterate: based on the results of the first experiments we may have to completely revise our models. The iterative data mining process cycle has been best illustrated by selecting the appropriate graph weighting scheme for the data mining problem: link prediction, clustering or recommendation.

Chapter 8

References

[1] D. Achlioptas and F. McSherry. On spectral learning of mixtures of distributions. In *Proceedings of the 18th Annual Conference on Learning Theory (COLT)*, pages 458–469, 2005.

[2] D. Achlioptas, A. Fiat, A. R. Karlin, and F. McSherry. Web search via hub synthesis. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 500–509, 2001.

[3] L. Adamic and E. Adar. Friends and neighbors on the Web. *Social Networks*, 25(3): 211–230, 2003.

[4] L. A. Adamic and N. Glance. The political blogosphere and the 2004 u.s. election: divided they blog. In *LinkKDD '05: Proceedings of the 3rd international workshop on Link discovery*, pages 36–43, New York, NY, USA, 2005. ACM.

[5] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. MIT Press, 1996.

[6] W. Aiello, F. Chung, and L. Lu. A random graph model for massive graphs. In *Proceedings of the 32th ACM Symposium on Theory of Computing (STOC)*, pages 171–180, 2000.

[7] R. Albert, H. Jeong, and A.-L. Barabási. Diameter of the world wide web. *Nature*, 401: 130–131, 1999.

[8] C. J. Alpert and A. B. Kahng. Multiway partitioning via geometric embeddings, orderings, and dynamic programming. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 14(11):1342–1358, 1995.

[9] C. J. Alpert and A. B. Kahng. Recent directions in netlist partitioning: a survey. *Integr. VLSI J.*, 19(1-2):1–81, 1995.

[10] C. J. Alpert and S.-Z. Yao. Spectral partitioning: the more eigenvectors, the better. In *DAC '95: Proceedings of the 32nd ACM/IEEE conference on Design automation*, pages 195–200, New York, NY, USA, 1995. ACM Press.

[11] E. Amitay, D. Carmel, A. Darlow, R. Lempel, and A. Soffer. The Connectivity Sonar: Detecting site functionality by structural patterns. In *Proceedings of the 14th ACM Conference on Hypertext and Hypermedia (HT)*, pages 38–47, Nottingham, United Kingdom, 2003.

[12] W.-H. Au, K. C. C. Chan, and X. Yao. A novel evolutionary data mining algorithm with applications to churn prediction. *IEEE Trans. Evolutionary Computation*, 7(6):532–545, 2003.

[13] Y. Azar, A. Fiat, A. R. Karlin, F. McSherry, and J. Saia. Spectral analysis of data. In *Proceedings of the 33rd ACM Symposium on Theory of Computing (STOC)*, pages 619–626, 2001.

[14] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 44–54, New York, NY, USA, 2006. ACM Press.

[15] A. Barabási. *Linked*. Perseus Publishing, 2002.

[16] A.-L. Barabási and R. Albert. Dynamics of complex system: Scaling laws for the period of boolean networks. *Physical Rewiew Letters*, 84:5660–5663, 2000.

[17] A.-L. Barabási, R. Albert, and H. Jeong. Mean-field theory for scale-free random network. *Physica A*, 272:173–187, 1999.

[18] A.-L. Barabási, R. Albert, and H. Jeong. Scale-free characteristics of random networks: the topology of the word-wide web. *Physica A*, 281:69–77, 2000.

[19] E. R. Barnes. An algorithm for partitioning the nodes of a graph. *SIAM Journal on Algebraic and Discrete Methods*, 3(4):541–550, Dec. 1982.

[20] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. Baeza-Yates. Link-based characterization and detection of web spam. In *Proceedings of the 2nd International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2006.

[21] E. Beltrami. Sulle funzioni bilineari. *Giornale di Matematiche ad Uso degli Studenti Della Universita*, (11):98–106, 1873.

[22] A. A. Benczúr, K. Csalogány, M. Kurucz, A. Lukács, and L. Lukács. Sociodemographic exploration of telecom communities. In *NSF US-Hungarian Workshop on Large Scale Random Graphs Methods for Modeling Mesoscopic Behavior in Biological and Physical Systems*, 2006.

[23] A. A. Benczúr, K. Csalogány, and T. Sarlós. Link-based similarity search to fight web spam. In *Proceedings of the 2nd International Workshop on Adversarial Information Retrieval on the Web (AIRWeb), held in conjunction with SIGIR2006*, 2006.

[24] A. A. Benczúr, K. Csalogány, M. Kurucz, A. Lukács, and L. Lukács. Telephone call network data mining: A survey with experiments. In B. Bollobás, R. Kozma, and D. Miklós, editors, *Handbook of Large-Scale Random Networks*, volume 18 of *Bolyai Society Mathematical Studies*. Springer Verlag in conjunction with the Bolyai Mathematical Society of Budapest, 2009.

[25] J. Bennett and S. Lanning. The netflix prize. In *KDD Cup and Workshop in conjunction with KDD 2007*, 2007.

[26] M. W. Berry. SVDPACK: A Fortran-77 software library for the sparse singular value decomposition. Technical report, University of Tennessee, Knoxville, TN, USA, 1992.

[27] M. W. Berry, S. T. Dumais, and G. W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595, 1995.

[28] D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 46–54, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[29] P. Boldi, B. Codenotti, M. Santini, and S. Vigna. Ubicrawler: A scalable fully distributed web crawler. *Software: Practice & Experience*, 34(8):721–726, 2004.

[30] B. Bollobás. *Random Graphs*. Academic Press, 1985.

[31] B. Bollobás, O. Riordan, J. Spencer, and G. Tusnády. The degree sequence of a scale-free random graph process. *Random Struct. Algorithms*, 18(3):279–290, 2001.

[32] A. Borodin, G. O. Roberts, J. S. Rosenthal, and P. Tsaparas. Finding authorities and hubs from link structures on the world wide web. In *Proceedings of the 10th World Wide Web Conference (WWW)*, pages 415–429, 2001.

[33] M. Brand. Incremental singular value decomposition of uncertain data with missing values. In *ECCV (1)*, pages 707–720, 2002.

[34] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[35] A. Z. Broder. On the Resemblance and Containment of Documents. In *Proceedings of the Compression and Complexity of Sequences (SEQUENCES'97)*, pages 21–29, 1997.

[36] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630–659, 2000.

[37] A. Z. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. In *Proceedings of the 9th World Wide Web Conference (WWW)*, pages 309–320. North-Holland Publishing Co., 2000.

[38] S. Burer and R. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95(2):329–357, 2003.

[39] J. Canny. Collaborative filtering with privacy via factor analysis. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 238–245, New York, NY, USA, 2002. ACM Press.

[40] P. K. Chan, M. D. F. Schlag, and J. Y. Zien. Spectral k-way ratio-cut partitioning and clustering. In *DAC '93: Proceedings of the 30th international conference on Design automation*, pages 749–754, New York, NY, USA, 1993. ACM Press.

[41] M. Charikar. Similarity estimation techniques from rounding algorithms. *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, 2002.

[42] D. Cheng, R. Kannan, S. Vempala, and G. Wang. On a recursive spectral algorithm for clustering from pairwise similarities. Technical report, MIT LCS Technical Report MIT-LCS-TR-906, 2003.

[43] D. Cheng, S. Vempala, R. Kannan, and G. Wang. A divide-and-merge methodology for clustering. In *PODS '05: Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 196–205, New York, NY, USA, 2005. ACM Press.

[44] F. Chung and L. Lu. The average distances in random graphs with given expected degrees. *Proceedings of the National Academy of Sciences of the United States of America*, 99(25):15879–15882, 2002.

[45] F. Chung, L. Lu, and V. Vu. Eigenvalues of random power law graphs. *Annals of Combinatorics*, 2003.

[46] F. Chung, L. Lu, and V. Vu. Spectra of random graphs with given expected degrees. *Proceedings of National Academy of Sciences*, 100:6313–6318, 2003.

[47] E. Cohen and D. D. Lewis. Approximating matrix multiplication for pattern recognition tasks. *Journal of Algorithms*, 30(2):211–252, 1999.

[48] G. Cormode, P. Indyk, N. Koudas, and S. Muthukrishnan. Fast mining of massive tabular data via approximate distance computations. In *ICDE '02: Proceedings of the 18th International Conference on Data Engineering*, page 605, Washington, DC, USA, 2002. IEEE Computer Society.

[49] K. C. Cox, S. G. Eick, G. J. Wills, and R. J. Brachman. Brief application description; visual data mining: Recognizing telephone calling fraud. *Data Min. Knowl. Discov.*, 1 (2):225–231, 1997.

[50] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[51] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm.

[52] I. Derényi, G. Palla, and T. Vicsek. Clique percolation in random networks. *Physical Review Letters*, 94:49–60, 2005.

[53] C. H. Q. Ding, X. He, and H. Zha. A spectral method to separate disconnected and nearly-disconnected web graph components. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 275–280, New York, NY, USA, 2001. ACM Press.

[54] C. H. Q. Ding, X. He, H. Zha, M. Gu, and H. D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 107–114, Washington, DC, USA, 2001. IEEE Computer Society.

[55] W. E. Donath and A. J. Hoffman. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17(5):420–425, Sept. 1973.

[56] Y. Dourisboure, F. Geraci, and M. Pellegrini. Extraction and classification of dense communities in the web. *Proceedings of the 16th international conference on World Wide Web*, pages 461–470, 2007.

[57] P. Drineas, I. Kerenidis, and P. Raghavan. Competitive recommendation systems. In *Proceedings of the 34th ACM Symposium on Theory of Computing (STOC)*, pages 82–90, 2002.

[58] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering large graphs via the singular value decomposition. *Machine Learning*, pages 9–33, 2004.

[59] P. Drineas, M. W. Mahoney, and R. Kannan. Fast Monte Carlo algorithms for matrices II: Computing a low rank approximation to a matrix. *SIAM Journal on Computing*, 36: 158–183, 2006.

[60] P. Drineas, M. W. Mahoney, and S. Muthukrishnan. Sampling algorithms for ℓ_2 regression and applications. In *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1127–1136, 2006.

[61] D. Duan, Y. Li, Y. Jin, and Z. Lu. Community mining on dynamic weighted directed graphs. In *Proceeding of the 1st ACM international workshop on Complex networks meet information & knowledge management*, pages 11–18. ACM, 2009.

[62] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1:211–218, 1936.

[63] P. Erdős and A. Rényi. On the evolution of random graph. *Math. Inst.*, 1960.

[64] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23 (98), 1973.

[65] G. Flake, S. Lawrence, and C. L. Giles. Efficient identification of web communities. In *Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 150–160, Boston, MA, August 20–23 2000.

[66] G. W. Flake, R. E. Tarjan, and K. Tsioutsiouliklis. Graph clustering and minimum cut trees. *Internet Mathematics*, 1(4):385–408, 2003.

[67] D. Fogaras. Singular Value Decomposition with Applications (in Hungarian). Technical report, Computer and Automation Research Institute of the Hungarian Academy of Sciences (MTA SZTAKI), 2002. `http://datamining.sztaki.hu/?q=en/en-publications`.

[68] D. Fogaras. Where to start browsing the web? In *Proceedings of the 3rd International Workshop on Innovative Internet Community Systems (I2CS)*, volume 2877/2003 of *Lecture Notes in Computer Science (LNCS)*, pages 65–79, Leipzig, Germany, June 2003. Springer-Verlag.

[69] D. Fogaras and B. Rácz. Practical Algorithms and Lower Bounds for Similarity Search in Massive Graphs. *IEEE Transactions on Knowledge and Data Engineering*, 19(5): 585–598, 2007. Preliminary version appeared at WWW 2005.

[70] D. Fogaras, B. Rácz, K. Csalogány, and T. Sarlós. Towards Scaling Fully Personalized PageRank: Algorithms, Lower Bounds, and Experiments. *Internet Mathematics*, 2(3): 333–358, 2005. Preliminary version from the first two authors appeared in WAW 2004.

[71] E. Frank and M. Hall. Additive Regression Applied to a Large-Scale Collaborative Filtering Problem. *AI 2008: Advances in Artificial Intelligence*, pages 435–446, 2008.

[72] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[73] A. Frieze, R. Kannan, and S. Vempala. Fast Monte-Carlo algorithms for finding low rank approximations. In *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 370–378, 1998.

[74] S. Funk. Netflix update: Try this at home., 2006. `http://sifter.org/ simon/journal/20061211.html`.

[75] K. R. Gabriel and S. Zamir. Lower rank approximation of matrices by least squares with any choice of weights. *Technometrics*, 21:489–498, 1979.

[76] Z. Ghahramani and M. I. Jordan. Supervised learning from incomplete data via an EM approach. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 120–127. Morgan Kaufmann Publishers, Inc., 1994.

[77] M. Girvan and M. E. Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA*, 99(12):7821–7826, June 2002.

[78] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Inf. Retr.*, 4(2):133–151, 2001.

[79] G. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, Apr. 1970.

[80] G. H. Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 1983.

[81] E. Gorny. Russian livejournal: National specifics in the development of a virtual community. pdf online, May 2004.

[82] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th International World Wide Web Conference (WWW)*, pages 403–412, 2004.

[83] A. Gunawardana and C. Meek. A unified approach to building hybrid recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, pages 117–124. ACM, 2009.

[84] D. Gupta, M. Digiovanni, H. Narita, and K. Goldberg. Jester 2.0 (poster abstract): evaluation of an new linear time collaborative filtering algorithm. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 291–292, New York, NY, USA, 1999. ACM Press.

[85] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Web content categorization using link information. Technical report, Stanford University, 2006–2007.

[86] L. W. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 11(9):1074–1085, 1992.

[87] H. O. Hartley. Maximum likelihood estimation from incomplete data. *Biometrics*, 14: 174–194, 1958.

[88] T. Hastie, R. Tibshirani, G. Sherlock, M. Eisen, O. Alter, D. Botstein, and P. Brown. Imputing missing data for gene expression arrays. Technical report, Department of Statistics, Stanford University, 2000.

[89] D. Hull. Improving text retrieval for the routing problem using latent semantic indexing. In W. B. Croft and C. J. van Rijsbergen, editors, *Proceedings of the 17th Annual International Conference on Research and Development in Information Retrieval*, pages 282–291, London, UK, July 1994. Springer Verlag.

[90] G. Jeh and J. Widom. SimRank: A measure of structural-context similarity. In *Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 538–543, 2002.

[91] G. Jeh and J. Widom. Scaling personalized web search. In *Proceedings of the 12th World Wide Web Conference (WWW)*, pages 271–279. ACM Press, 2003.

[92] R. Kannan, S. Vempala, and A. Vetta. On clusterings — good, bad and spectral. In *IEEE:2000:ASF*, pages 367–377, 2000.

[93] R. Kannan, H. Salmasian, and S. Vempala. The spectral method for general mixture models. In *Proceedings of the 18th Annual Conference on Learning Theory (COLT)*, pages 444–457, 2005.

[94] T. Kato, H. Kashima, M. Sugiyama, and K. Asai. Conic Programming for Multi-Task Learning. *IEEE Transactions on Knowledge and Data Engineering*, 2009.

[95] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1): 39–43, 1953.

[96] J. Kleinberg. Navigation in a small world. *Nature*, page 845, 2000.

[97] J. Kleinberg. The Small-World Phenomenon: An Algorithmic Perspective. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, 2000.

[98] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[99] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[100] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the web graph. In *Proceedings of the 41st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1–10, 2000.

[101] R. Kumar, J. Novak, P. Raghavan, and A. Tomkins. Structure and evolution of blogspace. *Commun. ACM*, 47(12):35–39, 2004.

[102] M. Kurucz and A. Benczúr. Geographically Organized Small Communities and the Hardness of Clustering Social Networks. *Annals of Information Systems*, pages 177–199, 2010.

[103] M. Kurucz, A. A. Benczúr, and K. Csalogány. Methods for large scale svd with missing values. In *KDD Cup and Workshop in conjunction with KDD 2007*, 2007.

[104] M. Kurucz, A. A. Benczúr, K. Csalogány, and L. Lukács. Spectral clustering in telephone call graphs. In *WebKDD/SNAKDD Workshop 2007 in conjunction with KDD 2007*, 2007.

[105] M. Kurucz, A. A. Benczúr, T. Kiss, I. Nagy, A. Szabó, and B. Torma. Who rated what: a combination of svd, correlation and frequent sequence mining. In *KDD Cup and Workshop in conjunction with KDD 2007*, 2007.

[106] M. Kurucz, A. Benczúr, and A. Pereszlényi. Large-Scale Principal Component Analysis on LiveJournal Friends Network. In *Workshop on Social Network Mining and Analysis Held in conjunction with The 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2008)*, 2008.

[107] S. Lalas, N. Ampazis, A. Tsakonas, G. Dounias, and K. Vemmos. Modeling Stroke Diagnosis with the Use of Intelligent Techniques. *Artificial Intelligence: Theories, Models and Applications*, pages 352–358, 2008.

[108] K. Lang. Finding good nearly balanced cuts in power law graphs. Technical report, Yahoo! Inc, 2004.

[109] K. Lang. Fixing two weaknesses of the spectral method. In *NIPS '05: Advances in Neural Information Processing Systems*, volume 18, Vancouver Canada, 2005.

[110] R. Lempel and S. Moran. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. *Computer Networks*, 33(1–6):387–401, 2000.

[111] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proceedings of the 12th Conference on Information and Knowledge Management (CIKM)*, pages 556–559, 2003.

[112] W. Lu, J. Janssen, E. Milios, and N. Japkowicz. Node similarity in networked information spaces. In *Proceedings of the Conference of the Centre for Advanced Studies on Collaborative research*, page 11, 2001.

[113] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *Int. J. Comput. Vision*, 43(1):7–27, 2001.

[114] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, England, 2000.

[115] J. Martinez, J. Huang, R. Burghardt, R. Barhoumi, and R. Carroll. Use of multiple singular value decompositions to analyze complex intracellular calcium ion signals. *The annals of applied statistics*, 3(4):1467, 2009.

[116] F. McSherry. Spectral partitioning of random graphs. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 529–537, 2001.

[117] M. Meila and J. Shi. A random walks view of spectral segmentation. In *AISTATS*, 2001.

[118] S. Milgram. The small world problem. *Psychology Today*, 2(1):60–67, 1967.

[119] A. A. Nanavati, S. Gurumurthy, G. Das, D. Chakraborty, K. Dasgupta, S. Mukherjea, and A. Joshi. On the structural properties of massive telecom graphs: Findings and implications. In *CIKM*, 2006.

[120] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *2009 30th IEEE Symposium on Security and Privacy*, pages 173–187. IEEE, 2009.

[121] M. Newman. The Structure and Function of Complex Networks. *SIAM Review*, 45(2): 167–256, 2003.

[122] M. Newman. Detecting community structure in networks. *The European Physical Journal B - Condensed Matter*, 38(2):321–330, March 2004.

[123] M. Newman. Clustering and preferential attachment in growing networks. *Physical Review E*, 64(2):25102, 2001.

[124] A. Y. Ng, A. X. Zheng, and M. I. Jordan. Link analysis, eigenvectors and stability. In *Proc. Int. Joint Conf. Artificial Intelligence, Seattle, WA*, August 2001.

[125] J. Onnela, J. Saramaki, J. Hyvonen, G. Szabó, M. de Menezes, K. Kaski, A. Barabási, and J. Kertész. Analysis of a large-scale weighted network of one-to-one human communication. *New Journal of Physics*, 9(6):179, 2007.

[126] J. Onnela, J. Saramaki, J. Hyvonen, G. Szabó, D. Lazer, K. Kaski, J. Kertész, and A. Barabási. Structure and tie strengths in mobile communication networks. *Proceedings of the National Academy of Sciences*, 104(18):7332, 2007.

[127] Open Directory Project (ODP). Open Directory Project (ODP). `http://www.dmoz.org`.

[128] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford University, 1998.

[129] G. Palla, A. Barabási, and T. Vicsek. Quantifying social group evolution. *Nature*, 446 (7136):664–667, 2007.

[130] G. Palla, D. Ábel, I. J. Farkas, P. Pollner, I. Derényi, and T. Vicsek. K-clique percolation and clustering. *Handbook of Large-Scale Random Networks*, 18:369–408, 2008.

[131] R. Pan and M. Scholz. Mind the gaps: weighting the unknown in large-scale one-class collaborative filtering. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 667–676. ACM, 2009.

[132] R. Pan, Y. Zhou, B. Cao, N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *Eighth IEEE International Conference on Data Mining, 2008. ICDM'08*, pages 502–511, 2008.

[133] C. H. Papadimitriou, H. Tamaki, P. Raghavan, and S. Vempala. Latent semantic indexing: A probabilistic analysis. *Journal of Computer and System Sciences*, 61(2):217–235, 2000.

[134] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD Cup and Workshop*, volume 2007, pages 5–8, 2007.

[135] K. Pearson. LIII. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine Series 6*, 2(11):559–572, 1901.

[136] D. M. Pennock, C. L. Giles, G. W. Flake, S. Lawrence, and E. Glover. Winners don't take all: A model of web link accumulation. *Proceedings of the National Academy of Sciences*, 99:5207–5211, April 2000.

[137] R. Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(03):406–413, 1955.

[138] M. H. Pryor. The effects of singular value decomposition on collaborative filtering. Technical report, Dartmouth College, Hanover, NH, USA, 1998.

[139] S. Rendle and L. Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 251–258. ACM, 2008.

[140] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–70, New York, NY, USA, 2002. ACM Press.

[141] A. Ruhe. Numerical computation of principal components when several observations are missing. Technical report, UMINF-48, Umeå, Sweden, 1974.

[142] T. Sarlós. Improved approximation algorithms for large matrices via random projections. In *Proceedings of the 47th IEEE Symposium on Foundations of Computer Science (FOCS)*, 2006.

[143] T. Sarlós, A. A. Benczúr, K. Csalogány, D. Fogaras, and B. Rácz. To randomize or not to randomize: Space optimal summaries for hyperlink analysis. In *Proceedings of the 15th International World Wide Web Conference (WWW)*, pages 297–306, 2006.

[144] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems–a case study. In *ACM WebKDD Workshop*, 2000.

[145] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM Press.

[146] J. Scott. *Social Network Analysis: A Handbook*. Sage Publications, 2000.

[147] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2000.

[148] X. Shi, B. Tseng, and L. Adamic. Looking at the blogosphere topology through different lenses. In *Proceedings Int. Conf. on Weblogs and Social Media (ICWSM-2007)*, 2007.

[149] M. Shiga, I. Takigawa, and H. Mamitsuka. A spectral clustering approach to optimally combining numerical vectors with a modular network. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 647–656, New York, NY, USA, 2007. ACM.

[150] N. Srebro and T. Jaakkola. Weighted low-rank approximations. In T. Fawcett and N. Mishra, editors, *ICML*, pages 720–727. AAAI Press, 2003.

[151] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Investigation of various matrix factorization methods for large recommender systems. In *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*, pages 1–8. ACM, 2008.

[152] G. Takács, I. Pilászy, B. Németh, and D. Tikk. A unified approach of factor models and neighbor based methods for large recommender systems. In *Applications of Digital Information and Web Technologies, 2008. ICADIWT 2008. First International Conference on the*, pages 186–191. IEEE, 2008.

[153] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Scalable collaborative filtering approaches for large recommender systems. *The Journal of Machine Learning Research*, 10:623–656, 2009.

[154] O. G. Troyanskaya, M. Cantor, G. Sherlock, P. O. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):520–525, 2001.

[155] U. von Luxburg, O. Bousquet, and M. Belkin. Limits of spectral clustering. pages 857–864, Cambridge, MA, 2005. MIT Press.

[156] D. J. Watts and S. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[157] C.-P. Wei and I.-T. Chiu. Turning telecommunications call details to churn prediction: a data mining approach. *Expert Syst. Appl.*, 23(2):103–112, 2002.

[158] Y. Weiss. Segmentation using eigenvectors: A unifying view. In *ICCV (2)*, pages 975–982, 1999.

[159] G. J. Wills. NicheWorks — interactive visualization of very large graphs. *Journal of Computational and Graphical Statistics*, 8(2):190–212, 1999.

[160] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, second edition, June 2005.

[161] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger. Scan: a structural clustering algorithm for networks. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 824–833, New York, NY, USA, 2007. ACM Press.

[162] J. Yeh and M. Wu. Recommendation Based on Latent Topics and Social Network Analysis. In *2010 Second International Conference on Computer Engineering and Applications*, pages 209–213. IEEE, 2010.

[163] K. Yu, J. Lafferty, S. Zhu, and Y. Gong. Large-scale collaborative prediction using a nonparametric random effects model. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1185–1192. ACM, 2009.

[164] K. Yu, S. Zhu, J. Lafferty, and Y. Gong. Fast nonparametric matrix factorization for large-scale collaborative filtering. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 211–218. ACM, 2009.

[165] P. Zakharov. Structure of livejournal social network. In *Proceedings of SPIE Volume 6601, Noise and Stochastics in Complex Systems and Finance*, 2007.

[166] H. Zha, X. He, C. H. Q. Ding, M. Gu, and H. D. Simon. Spectral relaxation for k-means clustering. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *NIPS*, pages 1057–1064. MIT Press, 2001.

[167] H. Zhang and R. Dantu. Discovery of Social Groups Using Call Detail Records. In *On the Move to Meaningful Internet Systems: OTM 2008 Workshops*, pages 489–498. Springer, 2010.

[168] J. Zhang, M. S. Ackerman, and L. Adamic. Expertise networks in online communities: structure and algorithms. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 221–230, New York, NY, USA, 2007. ACM Press.

[169] S. Zhang, W. Wang, J. Ford, F. Makedon, and J. Pearlman. Using singular value decomposition approximation for collaborative filtering. In *CEC '05: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology (CEC'05)*, pages 257–264, Washington, DC, USA, 2005. IEEE Computer Society.

[170] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan. Large-scale parallel collaborative filtering for the netflix prize. *Algorithmic Aspects in Information and Management*, pages 337–348, 2008.